Int J Parallel Prog (2014) 42:853–872
DOI 10.1007/s10766-013-0274-5

An Optimization-Based Scheme for Efficient Virtual Machine Placement

Fei Song · Daochao Huang · Huachun Zhou · Hongke Zhang · Ilsun You

Received: 19 April 2013 / Accepted: 7 September 2013 / Published online: 27 September 2013

Abstract According to the important methodology of convex optimization theory, the energy-efficient and scalability problems of modern data centers are studied. Then a novel virtual machine (VM) placement scheme is proposed for solving these problems in large scale. Firstly, by referring the definition of VM placement fairness and utility function, the basic algorithm of VM placement which fulfills server constraints of physical machines is discussed. Then, we abstract the VM placement as an optimization problem which considers the inherent dependencies and traffic between VMs. By given the structural differences of recently proposed data center architectures, we further investigate a comparative analysis on the impact of the network architectures, server constraints and application dependencies on the potential performance gain of optimization-based VM placement. Comparing with the existing schemes, the

F. Song · D. Huang · H. Zhou · H. Zhang
School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, People's Republic of China
e-mail: fsong@bjtu.edu.cn

D. Huang
e-mail: dchhuang@bjtu.edu.cn

H. Zhou
e-mail: hchzhou@bjtu.edu.cn

H. Zhang
e-mail: hkzhang@bjtu.edu.cn

F. Song · D. Huang · H. Zhou · H. Zhang
National Engineering Lab for Next Generation Internet Interconnection Devices, Beijing 100044, People's Republic of China

I. You (✉)
School of Information Science, Korean Bible University, Seoul, South Korea
e-mail: ilsunu@gmail.com

performance improvements are illustrated from multiple perspectives, such as reducing the number of physical machines deployment, decreasing communication cost between VMs, improving energy-efficient and scalability of data centers.

Keywords Optimization theory · Virtual machine placement · Virtualization · Data center

1 Introduction

The functions of recent modern virtualized data center could be treated as an infrastructure for Internet applications which requires large-scale data processing services, such as search engines, public medical services, telecommunications, sensor networking, social networking, data mining, information storage, electronic commerce, as well as some more general services such as software as a Service (SaaS), platform as a Service (PaaS), infrastructure as a Service (IaaS), grid computing and cloud computing, etc. With the development of the large scale cloud computing [1–3] and communication-based data center [4–6], bandwidth and other resources requirements in virtual machine (VM) [7] have increased dramatically, which has been attracting extensive interests in underlying network architecture scalability research [8–11], these studies are trying to reduce data center network costs by increasing the degree of network connectivity and using dynamic routing protocols to balance transmission workloads. In such background, with the cooling energy demand and costs augment, power consumption management has to be considered. Following this point, researchers have proposed some green computing concepts and suggestions [12,13], the main purpose for these work is to reduce energy consumption in massive data processing to save costs. These methods can produce obvious effects on the data center server resources, and reduce energy consumption by shutting down or hibernating servers with low workloads, which facilitate us to find new mechanisms to solve the problem.

The motivation of our work is to handle the data center scalability and energy consumption reduction problems from the perspective of optimizing virtual machine deployment. Virtual machine placement (VMP) is to establish mapping relationships between virtual machine and physical machine (PM) so as to select the most suitable hosting carrier for each virtual machine. It has been widely discussed in multiple directions [14,15]. The basic process takes into account of the virtual machine hardware, resource requirements, and anticipated deployment targets (such as maximizing utilization of available resources, saving energy, etc.). Existing virtual machine placement tools include VMware Capacity Planner [16], IBM WebSphere CloudBurst [17], Novell PlateSpin Recon [18] and Lanamark Suite [19], etc. These tools, to a certain extent, automatically distribute the virtual machines in data center based on the requirements of the CPU, physical memory and energy consumption. Since these tools do not consider the demands of application awareness, it may lead to application associated virtual machines being placed in physical machines that require long distance in network communication, which increases the burden of data transfer within large-scale data centers.

The main contribution of this paper is to propose an optimization-based algorithm for virtual machines placements. The algorithm considers both constraints of servers and the dependency relationships between virtual machines and multiple levels applications. We try to place a virtual machine into the most appropriate physical host. By meeting the conditions of the server-side constraints, our objective is to minimize the number of physical hosts, improve the scalability and reduce the energy consumption. The evaluation results show that the algorithm can adjust the allocation of resources and reduce the transmission traffic of data center in a short period.

The structure of this paper is organized as follows: Sect. 1 introduces the background of this area and the main research contents; Sect. 2 describes the relevant virtual machine placement tools and schemes; While Sect. 3 derives the utility function meeting server-side constraints and the optimization-based utility function expression; In Sect. 4, the experimental results of the analysis under different data center network architecture are presented based on the proposed virtual machines placement algorithm; Sect. 5 summarizes this paper and give the future work.

2 Related Work

Existing virtual machine placement algorithms can be divided into two categories based on the placement target: one is based on the energy consumption, these methods mainly consider the server-side constraints; the other is based on application Quality of Service (QoS), aiming to maximize the use of the resources allocated to applications. Several common virtual machine placements algorithms include Constraint Programming [20,21], Bin Packing Stochastic [22], Integer Programming [23], Genetic Algorithm [24,25], some of which involves a virtual machine dynamical migration, while the others simply consider the static cases.

The requirements of virtual machines autonomous management satisfying the quality of service can be modeled as a Constraint Programming Problem. Van et al. [20] proposed a constraint programming virtual machine placement algorithm that can reduce the number of physical machines and maximize the global utility function under the constraints of achieving quality of service service-level agreement (SLA) and operating costs. In this paper, the utility function maps the application's current status (workload, resource capacity, SLA, etc.) to a constant value, which reflects the application's satisfaction degree. Virtual machine placement algorithm based on constraint planning can be seen as a compound process including both local and global decision-making processes. The first stage is related to application environment, while the second stage calculates the maximum of the utility function based on the inputs of local decisions for all applications. In local decision-making stage, the application locally maps quality of service to the utility value using the fixed QoS (service-level) utility function, while it maps resource capacity to the utility value as well using the dynamic quality of resources (resource-level) utility function, and ultimately quantify the resource level value as the global decision-making input for each iteration. In the global decision-making stage, the system firstly need to find each application's virtual machine placement vector and then get the maximum value of the global utility function, followed by deploying corresponding virtual machines to physical ones and minimizing the number of activated physical machines. In [20], it is important to select the appropriate application weights. If it is set by the system administrator, the situation could be maintained; if it is achieved through automated virtual machine management, then extra checks might be needed to find out whether the application requires resources. Besides that, the global optimization function of the algorithm depends on a given network architecture, limiting its scope of application.

Meng et al. [21] proposes an effective solution for data center scalability. The core part of this work is TVMPP (Traffic-aware VM Placement Problem) algorithm. Such algorithm is able to reasonably place virtual machines that require large number of data exchanges and high network bandwidth, greatly reduce the bandwidth consumption within the data center and improve the scalability. However, TVMPP algorithm only considered the network constraints (such as bandwidth) of data center and neglects the server-side constraints, which may lead to the virtual machine workload exceeding the capacity of physical machines, hot spots and other critical issues.

For the basic idea of Bin Packing Problem, physical machines can be seen as a "box", and virtual machines can be seen as an object into the box. Bobroff et al. [22] introduced the Measure-Forcast-Remap (MFR) algorithm, which contain three steps: a) the formation of the expression pattern of past demand; b) predicting future needs based on past demand patterns; c) mapping or re-mapping the virtual machine to the suitable physical machine. MFR algorithm uses first-fit approximation to find a virtual machine to physical machine mapping, and the migration of virtual machine is also considered. The algorithm does not take the resource capacity constraints, which means there may be interference, in term of, the virtual machines are placed into the same physical machine. In addition, the algorithm highly depends on the effectiveness of prediction algorithms, if the forecast is not accurate enough, SLA breach of contract will exceed the tolerable range.

Chaisiri et al. [23] proposed a Statistical Integer Planning (SIP) algorithm. This algorithm achieves the function of resources allocation via three steps as well: a) Reservation: Cloud providers pre-reserve resources for future allocation; b) Utilization: utilize the pre-reserved resources; c) On-demand: if the users' needs exceed the pre-reserved resources, additional resources based on demand can be allocated according to the paying fees. SIP algorithm categorizes the virtual machines based on the types of applications. Each application corresponds to a virtual machine, which minimize the cost in virtual machines deployment and meet the needs of users. The authors optimize the process of calculating the number of virtual machines at the reservation and on-demand stage. The calculating results will be changing dynamically with the distribution of demands. SIP algorithm is quite practical in independent demands or cost changes. It does not require frequent calculating. However, if the estimation is not accurate, users will pay more. In this paper, the authors did not give the mapping algorithms corresponding virtual machines to physical machines.

Based on genetic algorithm, virtual machine placement scheme proposed by Nakada in [24] is suitable for the situations where objective function changes dynamically. Agrawal et al. [25] designed a grouping genetic algorithm (GGA), which can be classified as a bin packing algorithm category. The GGA algorithm can not only achieve the effective "packaging" from virtual machines to physical machines, but also meet the corresponding constraints to avoid the bin packing algorithm defects.

These previous work provide many achievements in multiple angles and motivate us to settle the virtual machine placement issues with optimization theory.

3 Virtual Machine Placement Model

In this section, we present an optimization-based virtual machine placement model that enables data centers to systematically place virtual machines. The energy efficiency and scalability can be achieved as well. We consider both server capacity constraints and application multi-tier inherent dependencies in the process. The formulations are introduced firstly. The modeling work will be detailed in the following subsections.

3.1 Abstraction and Problem Formulation

Typically, in order to reduce the energy costs, most of the extant work focuses on only one specific aspect of management, such as minimizing power consumption, balancing thermal distribution, or maximizing resource usage. However, with many practical applications, minimizing the total energy consumption in a data center requires the formulation of a joint optimization problem. Therefore, we consider server-side constraints associated with application multi-tier inherent dependencies as a joint optimization model to solve energy efficiency and scalability problems. Due to multiple objectives may bring conflicts with each other, the definition of virtual machine placement fairness is given firstly.

Let $y = \left(y_p, p \in P\right)$ denote a VMP approach satisfying proportional fairness: assuming that (a) resources allocation policy is feasible, in other words, resources (such as CPU, physical memory, storage space, etc.) allocated to each virtual machine is less than the total capacity of the hosting physical machine. (b) for any other alternative resource allocation approach $\tilde{y} = \left(\tilde{y}_p, p \in P\right)$, the following condition is satisfied:

$$\sum_{p:p\in P} w_p \frac{\tilde{y}_p - y_p}{y_p} \le 0, \tag{1}$$

where w_p is the weight of server performance or its willingness to offer resources.

As we know, proportional fairness depends on the difference between two placement approaches, which means that both fairness and efficiency are integrated in one model. Based on this definition, the objective function of the overall utilities can be denoted as:

$$\sum_{p:p\in P} U_p\left(y_p(t)\right) = \begin{cases} w_p \log y_p(t) & \textit{if } \alpha_p = 1 \\ w_p \frac{y_p^{1-\alpha_p}(t)}{1-\alpha_p} & \textit{if } \alpha_p \neq 1 \end{cases}, \tag{2}$$

Here α_p is the indicator of fairness among a connection of physical machines. When $\alpha_p = 1$, the utility function of physical machine p is given by $U_p\left(y_p(t)\right) = w_p \log y_p(t)$. Then, maximizing the aggregate utilities of all physical machines in data center can be given by:

Table 1 Notation used in our model

Notation	Description
$V = \{v_1, v_2, \ldots, v_n\}$	Set of virtual machines, element $v_i \in V, i = 1, 2, \ldots, n$
$P = \{p_1, p_2, \ldots, p_m\}$	Set of physical machines, element $p_k \in P, k = 1, 2, 3, \ldots, m$
$V(p)$	Set of virtual machines hosted by physical machine $p \in P$
$y_p(t) = Load_p(t)$	The aggregate resource demands of virtual machines located at physical machine p
$x_{pv}(t) = Load_{pv}(t)$	The resource demands of virtual machine v hosted by physical machine p
$Capacity_p$	Capacity of physical machine p (e.g., CPU power, memory, storage)
$G = (V, E)$	Application dependency graph, where V is the set of virtual machines, E is the set of depending edges
$E = (v_i, v_j) : v_i, v_j \in V$	Set of depending edges, v_i and v_j are dependent with each other if any communication takes places between them
$W = \{w_{ij} = w(v_i, v_j) : v_i, v_j \in V\}$	Traffic demand for each edge, element $w_{ij} \in W$

$$\max \sum_{p \in P} w_p \log y_p(t). \tag{3}$$

Before giving the joint virtual machine placement model, some notations used in subsequent section are listed in Table 1.

3.2 Server-Side Constraints and Virtual Machine Placement

Multiple virtual machines can reside on the same host and each VM occupies part of physical resources. Let $Load_{pv}(t)$ denote the load of VM v which resides on PM $p \in P$, and $Load_p(t)$ denote the aggravate loads of PM $p \in P$, the following constraints must be satisfied: $Load_p(t) = \sum_{v:v \in V(p)} Load_{pv}(t)$. Here $Load(t)$ is defined as the d dimensional vector of load requirements of VM, when CPU, memory and storage are considered, $d = 3$, $Load(t)$ is given by

$$Load(t) = \left(Load^{CPU}(t), Load^{memory}(t), Load^{storage}(t)\right). \tag{4}$$

Further, we define $Capacity_p$ as the available server-side capacity on PM $p \in P$ regarding its CPU, memory, storage etc. To ensure that the total load on any physical machine is less than or equal to its capacity, the following formula must hold:

$$\sum_{v:v \in V(p)} Load_{pv}(t) \le Capacity_p. \tag{5}$$

We study the problem of placing VMs on a set of physical hosts here. Typically, tightly packing VMs onto a small number of servers and turning off other servers is an effective way to maximize machine utilization and reduce server energy consumption. However, concentrating workloads on a subset of the system resources can cause heat imbalances and create hot spots, which may impact cooling costs, shorten the server life and degrade the system performance. An effective strategy should maintain the tradeoffs between energy efficiency and fairness. To reflect these two considerations, in our analysis, proportional fairness model is utilized:

$$(\boldsymbol{P}_1): \max \sum_{p:p\in P} U_p\left(Load_p\left(t\right)\right) \tag{6}$$

$$\text{Subject to} \quad \sum_{v:v\in V(p)} Load_{pv}\left(t\right) = Load_p\left(t\right), \quad \forall p \in P \tag{7}$$

$$\sum_{v:v\in V(p)} Load_{pv}\left(t\right) \le Capacity_p, \quad \forall p \in P \tag{8}$$

$$\text{Over} \quad Load_{pv}\left(t\right) \ge 0,\, p \in P, \quad v \in V. \tag{9}$$

We call this physical machine utilization maximization problem $(\boldsymbol{P}_1)$. When only CPU, memory and storage are considered, $(\boldsymbol{P}_1)$ is equivalent to:

$$\left(\boldsymbol{P}_1'\right): \max \sum_{p:p\in P} U_p\left(Load_p^{CPU}\left(t\right) \times Load_p^{memory}\left(t\right) \times Load_p^{storage}\left(t\right)\right) \tag{10}$$

$$\text{Subject to} \begin{cases} \sum\limits_{v:v\in V(p)} Load_{pv}^{CPU}\left(t\right) = Load_p^{CPU}\left(t\right), \quad \forall p \in P \\ \sum\limits_{v:v\in V(p)} Load_{pv}^{memory}\left(t\right) = Load_p^{memory}\left(t\right), \quad \forall p \in P \\ \sum\limits_{v:v\in V(p)} Load_{pv}^{storage}\left(t\right) = Load_p^{storage}\left(t\right), \quad \forall p \in P \end{cases} \tag{11}$$

$$\begin{cases} \sum\limits_{v:v\in V(p)} Load_{pv}^{CPU}\left(t\right) \le Capacity_p^{CPU}, \quad \forall p \in P \\ \sum\limits_{v:v\in V(p)} Load_{pv}^{memory}\left(t\right) \le Capacity_p^{memory}, \quad \forall p \in P \\ \sum\limits_{v:v\in V(p)} Load_{pv}^{storage}\left(t\right) \le Capacity_p^{storage}, \quad \forall p \in P \end{cases} \tag{12}$$

$$\text{Over} \quad Load_{pv}\left(t\right) \ge 0,\, p \in P,\, v \in V. \tag{13}$$

In order to facilitate the subsequent derivation of the formula, let $y_p\left(t\right) = Load_p\left(t\right)$. To maximize the overall aggregate utilities of data center and obtain the optimal solution of $(\boldsymbol{P}_1)$, a Lagrange function is defined as:

$$\begin{aligned} L\left(x, y; \lambda, \eta\right) = & \sum_{p:p\in P} \left\{ U_p\left(y_p\left(t\right)\right) + \lambda_p \left(\sum_{v:v\in V(p)} x_{pv}\left(t\right) - y_p\left(t\right) \right) \right\} \\ & + \sum_{p:p\in P} \eta_p \left(Capacity_p - \sum_{v:v\in V(p)} x_{pv}\left(t\right) - \varepsilon_p^2 \right), \end{aligned} \tag{14}$$

Where both $\lambda = \left(\lambda_p, p \in P\right)$ and $\eta = \left(\eta_p, p \in P\right)$ are Lagrange multiplier vectors, $\varepsilon^2 = \left(\varepsilon_p^2, p \in P\right)$ is the relaxation factor vector. Let λ_{pv} denote the load requirements of virtual machine v. Let η_p be the available capacity of physical machine p. Let the total resources occupied by all the virtual machines residing on physical machine p be denoted by $\sum_{v:v\in V(p)} x_{pv}(t)$. $\varepsilon_p^2 \geq 0$ will indicate there are remaining resources exist on physical machine p.

According to (7), it is easy to obtain:

$$\begin{aligned} L(x, y; \lambda, \eta) = & \sum_{p:p\in P} \left\{U_p\left(y_p(t)\right) - \lambda_p y_p(t)\right\} \\ & + \sum_{p:p\in P} \sum_{v:v\in V(p)} x_{pv}(t)\left(\lambda_p - \eta_p\right) \\ & + \sum_{p:p\in P} \eta_p \left(Capacity_p - \varepsilon_p^2\right), \end{aligned} \tag{15}$$

Where $Capacity_p - \varepsilon_p^2$ indicates the occupied resources of physical machine p. To improve the physical machine utilization and reduce server energy consumption, tightly packing with fairness consideration is taken to place virtual machines, then, $\eta_p \left(Capacity_p - \varepsilon_p^2\right)$ can be considered as the gains of physical machine p from packing.

The dual problem of the objective function (15) is given by:

$$\begin{aligned} D(\eta) &= \max_{x,y,\lambda} L(x, y; \lambda, \eta) \\ &= \max_{\lambda} \sum_{p:p\in P} \left(\max_{y_p(t)} U_p\left(y_p(t)\right) - \lambda_p y_p(t)\right) \\ &\quad + \sum_{p:p\in P} \sum_{v:v\in V(p)} \max_{x_{pv}(t)} x_{pv}(t)\left(\lambda_p - \eta_p\right) \\ &\quad + \sum_{p:p\in P} \eta_p \left(Capacity_p - \varepsilon_p^2\right). \end{aligned} \tag{16}$$

Therefore, the dual problem of the basic virtual machine model ($\boldsymbol{P}_1$), denoted as ($\boldsymbol{D}_1$), is given by:

$$(\boldsymbol{D_1}) : \min D(\eta) \tag{17}$$

$$\text{Over} \qquad \eta_p \geq 0, p \in P. \tag{18}$$

Theorem 1 *The basic data center virtual machine model* ($\boldsymbol{P}_1$) *is a convex programming problem, the optimal resources distribution solution, denoted as* $x = \left(x_{pv}(t), p \in P, v \in V\right)$, *exists, but not unique, while the total load requirements of physical machines have unique solution.*

The proof of this theorem can be found in Appendix A.

Based on the virtual machine placement model (P_1), the derivation yields:

$$\frac{\partial L\left(x, y; \lambda, \eta\right)}{\partial y_p\left(t\right)} = 0, \tag{19}$$

then

$$y_p\left(t\right) = \frac{w_p}{\lambda_p}. \tag{20}$$

Following that we consider

$$\begin{aligned}\hat{L}\left(x; \lambda, \eta\right) = &\sum_{p:p\in P}\left\{w_p \log\left(\frac{w_p}{\lambda_p}\right) - w_p + \sum_{v:v\in V(p)} x_{pv}\left(t\right)\lambda_p\right\}\\ &+ \sum_{p:p\in P}\eta_p\left(Capacity_p - \sum_{v:v\in V(p)} x_{pv}\left(t\right) - \varepsilon_p^2\right).\end{aligned} \tag{21}$$

Let $\frac{\partial L(x;\lambda,\eta)}{\partial\lambda_p} = 0$, then

$$\lambda_p = \frac{w_p}{\sum_{v:v\in V(p)}} x_{pv}\left(t\right). \tag{22}$$

To simplify (21) with respect to (22), we get:

$$\begin{aligned}\hat{L}\left(x; \eta\right) = &\sum_{p:p\in P}\left\{w_p \log\left(\sum_{v:v\in V(p)} x_{pv}\left(t\right)\right)\right\}\\ &+ \sum_{p:p\in P}\eta_p\left(Capacity_p - \sum_{v:v\in V(p)} x_{pv}\left(t\right) - \varepsilon_p^2\right).\end{aligned} \tag{23}$$

Let $\frac{\partial L(x;\eta)}{\partial x_{pv}(t)} = 0$, and thus

$$y_p\left(t\right) = \sum_{v:v\in V(p)} x_{pv}\left(t\right) = \frac{w_p}{\eta_p}. \tag{24}$$

Comparing with (20), we get

$$\lambda_p = \eta_p. \tag{25}$$

It can be concluded that: when $\lambda_p = \eta_p$, the load requirements of virtual machines are equal to the availability capacity of physical machines, the optimal unique solution of ($\boldsymbol{P}_1$) can be obtained.

3.3 Optimization-Based Virtual Machine Placement

Assume that the data center administrator can obtain a dependency graph, $G(V, E)$, where V is the set of virtual machines and E is the set of edges defined as $E = (v_i, v_j)$: $v_i, v_j \in V$. v_i and v_j are dependent with each other if any communication takes places between them. Let $W(v_i, v_j)$ denote traffic demand for each edge which is directly proportional to the traffic transferred between v_i and v_j. At time t, the real traffic is given by $W_{ij}(t)$. Further, let $Load(v_i)$ be the vector of load requirements of virtual machine v_i, such as the vector of CPU, memory or storage. Let $Capacity(p_i)$ denote the available server-side capacity of physical machine p_i. Next, let $Distance(p_k, p_l)$ be the latency, delay or number of hops between physical machines p_k and p_l. Given that the indicator function of virtual machine placement is defined as:

$$I_{ik} = \begin{cases} 1 & if\ virtual\ machine\ v_i\ resides\ on\ physical\ machine\ p_k \\ 0 & otherwise \end{cases}. \tag{26}$$

From the above, $\sum_k^{|P|} I_{ik} = 1$ is satisfied, in other words, one virtual machine v_i must be located at a physical machine p_k. Similarly, let $I_{ik}^{jl} = I_{ik} \times I_{jl}$, $I_{ik}^{jl} = 1$ denote the situation where virtual machine v_i is assigned to p_k and v_j is assigned to p_l. Therefore, $I_{ik} + I_{jl} \leq 1 + I_{ik}^{jl}$.

Let $Cost(\cdot)$ be the communication cost function of placement approach (represented with $y_{p_k}(t)$) for the physical machine p_k at time t, for all $p_k \in P$. Then the total communication costs of physical machine can be defined as $\sum_{k=1}^{m} Cost\left(y_{p_k}(t)\right)$. In the calculation process, $Cost(\cdot)$ is represented as $Cost\left(y_{p_k}(t)\right) = w_{p_k} y_{p_k}(t)$, where w_{p_k} is the price of communication. The set of application dependencies for virtual machine v_i is denoted by $D(v_i)$. The aggregate communication traffic for physical machine p_k is expressed as:

$$y_{p_k}(t) = \sum_{\forall v_i \in V(p_k)} \sum_{\forall v_j \in D(v_i)} Distance(p_k, p_l) \times W(v_i, v_j) \times I_{ik}^{jl}. \tag{27}$$

From the perspective of network traffic, the total bandwidth requirement on any physical server should be less than or equal to its capacity. Therefore, the following formula must hold:

$$\sum_{i}^{|V|} \sum_{j, j \neq i}^{|V|} W_{ij}(t) \times I_{ik} \leq Bandwidth_k, \tag{28}$$

where the $Bandwidth_k$ is the bandwidth capacity of physical machine p_k. Then, the optimization-based objective function can be given by minimizing the aggregate communication costs among VMs deployed, which means the ($\boldsymbol{P}_2$) can be defined as follow:

$$(\boldsymbol{P}_2): \min \sum_{k=1}^{m} Cost\left(y_{p_k}(t)\right) \tag{29}$$

Subject to

$$\begin{aligned} y_{p_k}(t) &= \sum_{\forall v_i \in V(p_k)} \sum_{\forall v_j \in D(v_i)} Distance\,(p_k, p_l) \times W\left(v_i, v_j\right) \times I_{ik}^{jl} \\ &= \sum_{\forall v_i \in V(p_k)} \sum_{\forall v_j \in D(v_i)} Distance\,(p_k, p_l) \times W_{ij}(t) \times I_{ik}^{jl} \end{aligned} \tag{30}$$

$$\sum_{i}^{|V|} \sum_{j, j \neq i}^{|V|} W_{ij}(t) \times I_{ik} \leq Bandwidth_k \tag{31}$$

$$\text{Over} \qquad W_{ij}(t) \geq 0, i, j = 1, \ldots, n. \tag{32}$$

To facilitate the subsequent description, let

$$x_i(t) = \sum_{\forall v_j \in D(v_i)} Distance\,(p_k, p_l) \times W_{ij}(t) \times I_{ik}^{jl}. \tag{33}$$

Theorem 2 *The convex programming problem* ($\boldsymbol{P_2}$) *has unique optimal solution, that is, if*

$$R = \left\{ \boldsymbol{x} \middle| \sum\nolimits_{\forall v_i \in V(p_k)} x_i(t) \leq Bandwidth_k; x_i(t) \geq 0, i = 1, 2, \ldots, n \right\}, \tag{34}$$

then $R^* \neq \emptyset$. *The proof of this theorem can be found in Appendix B.*

Therefore, ($\boldsymbol{P}_2$) is equivalent to:

$$(\boldsymbol{P_2}) \begin{cases} \min \sum\limits_{k=1}^{m} Cost\left(y_{p_k}(t)\right) \\ Bandwidth_k - \sum\limits_{\forall v_i \in V(p_k)} x_i(t) \geq 0 \\ x_i(t) \geq 0, i = 1, 2, \ldots, n \end{cases} \cdot \tag{35}$$

Since the constraint functions of ($\boldsymbol{P_2}$) are linear, according to Karush-Kuhn-Tucker (*KKT*) [26] conditions. Assuming that

$$\begin{aligned} \varphi(\boldsymbol{x}, \boldsymbol{\mu}) &= \sum_{k=1}^{m} Cost\left(y_{p_k}(t)\right) \\ &- \sum_{k=1}^{m} \mu_k \left(Bandwidth_k - \sum_{i}^{|V|} \sum_{j, j \neq i}^{|V|} W_{ij}(t) \times I_{ik} \right) = \sum_{k=1}^{m} Cost\left(y_{p_k}(t)\right) \\ &- \sum_{k=1}^{m} \mu_k \left(Bandwidth_k - \sum\nolimits_{\forall v_i \in V(p_k)} x_i(t) \right). \end{aligned} \tag{36}$$

Then, the *KKT* conditions are:

$$\left\{\begin{array}{l}\frac{\varphi(\bar{x},\bar{\mu})}{\partial x_i(t)} = Cost_x\left(y_{p_k}(t)\right) + \bar{\mu}_k \geq 0 \\ \qquad x_i(t) \geq 0, i, j = 1, 2, \ldots, n; k, l = 1, 2, \ldots, m \\ \frac{\varphi(\bar{x},\bar{\mu})}{\partial \mu_k} = -\left(Bandwidth_k - \sum_{\forall v_i \in V(p_k)} x_i(t)\right) \leq 0, \quad \bar{\mu}_k \geq 0 \\ \frac{\varphi(\bar{x},\bar{\mu})}{\partial x_i(t)} x_i(t) = \left(Cost_x\left(y_{p_k}(t)\right) + \bar{\mu}_k\right) x_i(t) = 0 \\ \qquad i, j = 1, 2, \ldots, n; k, l = 1, 2, \ldots, m \\ \bar{\mu}_k \frac{\varphi(\bar{x},\bar{\mu})}{\partial \mu_k} = -\bar{\mu}_k\left(Bandwidth_k - \sum_{\forall v_i \in V(p_k)} x_i(t)\right) = 0 \end{array}\right. . \tag{37}$$

4 Performance Validation

In order to validate the performance of our virtual machine placement algorithm in different scenarios, we consider four widely-used architectures of data center network: Tree, VL2, Fat-Tree and BCube. By taking into account communication approaches between two servers in different structures, the expressions are given in Appendix C.

4.1 Scenario One: Performance Comparisons in Histogram

By using the four architectures mentioned above, we compared our optimized placement algorithm with random placement algorithm and bin-packing placement algorithm firstly. In order to unify the target of two kinds of optimization problem, a fuzzy mechanism is adopted. Objective value is used to show the differences: the smaller objective value indicates the better performance. In the simulation, we generated the Tree, VL2, Fat-Tree and BCube topologies. The number of servers is set to 16. Specific parameter and operation settings are: traffic demands of virtual machines meet the normally distribution function with different parameters (mean and variance value); the interdependencies among applications are varying randomly; the capacity of physical machines exceeds the total load demands of virtual machines; the number of virtual machines is set to 16 as well. Simulation results are illustrated with the average value of 1000 operations.

Figure 1 shows the comparison results between random algorithm and the optimization-based algorithm, where the horizontal axis of the figure stands for the average traffic in virtual machine. For example, 0.4 represents that communication traffic between virtual machines meets the normal distribution with 0.4 as mean (and 0.1 as variance for all tests in this scenario). The bars in histogram are used to display the objective value for each algorithm in multiple data center architectures. Since the overlap occurs, the top part of each bar indicates the potential enhancement from the random placement model to the optimization-based placement model. A reminder is that the objective value should be minimized. Figure 1 also shows that, comparing with other three network architectures (Tree, VL2 and Fat-Tree), the communication costs of the BCube benefited most by using our placement algorithm.

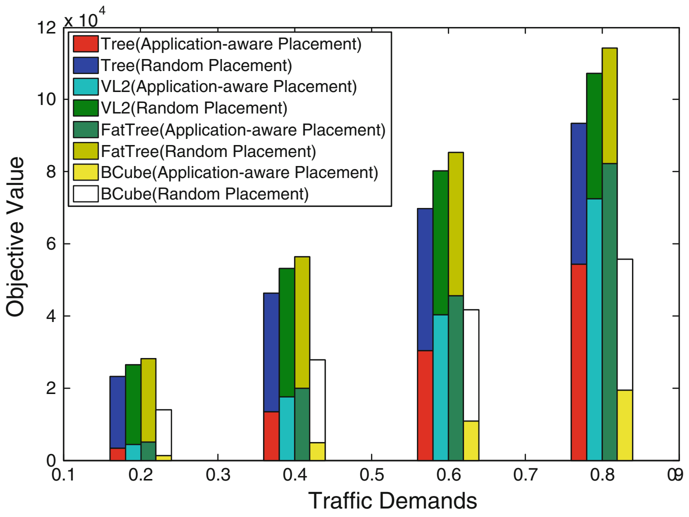

Fig. 1 Optimization-based VMP versus Random VMP

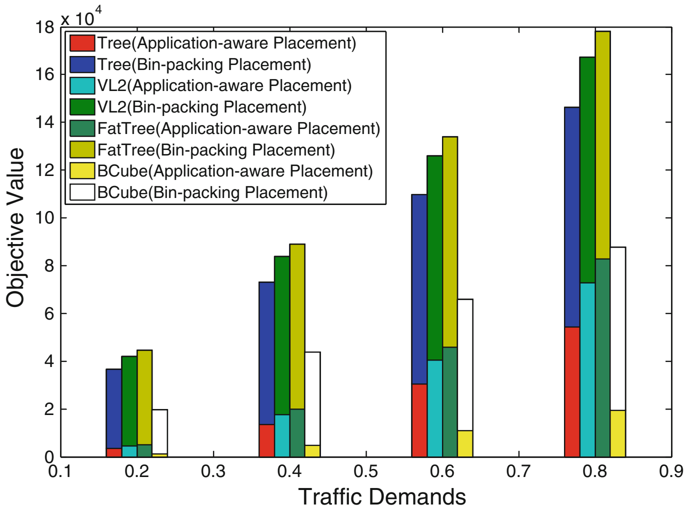

Fig. 2 Optimization-based VMP versus Bin-packing VMP

Figure 2 compares the objective values of traditional bin-packing algorithm with our optimization-based algorithm. The results can lead us to summarize the similar standpoint which is the communication cost of the new scheme is much lower than that of the traditional bin-packing scheme. BCube is still the "best current practice", in terms of performance improvement in this scenario. Therefore in the next scenario, we will be focusing on the other three architectures.

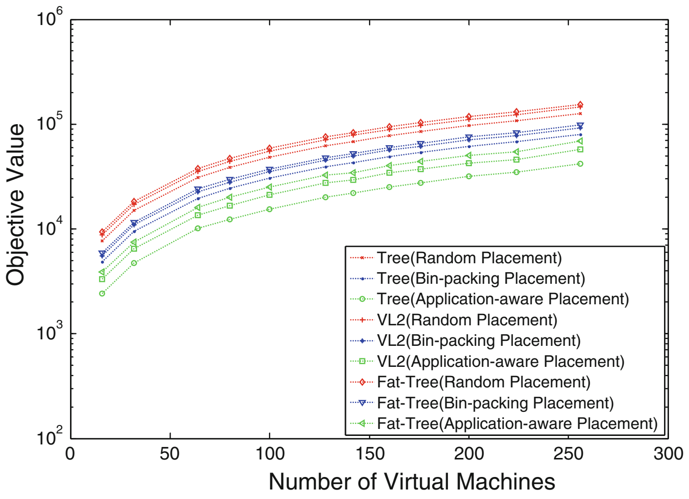

Fig. 3 The comparisons of three VMP algorithms

4.2 Scenario Two: Performance Comparisons in Curve

Figure 3 shows the performance comparison results of the three optional placement algorithms under different network architectures (Tree, VL2 and Fat-Tree) with the number of virtual machines increasing. The simulation parameters and operation settings are as follow: the traffic demands among virtual machines meet the evenly distribution in range [0.1, 0.9]; randomly set the interdependencies among applications; the capacity of physical machines exceeds the total load demands of virtual machines; the number of virtual machines is set to 16, 32, 64, 80, 100, 128, 142, 160, 176, 200, 224 and 256, respectively. In order to avoid randomness in selecting random values (virtual machine load or physical capacity of the machine), the results showed here still selected the average value of 1000 runs. The horizontal axis in Fig. 3 stands for the number of virtual machines, and the vertical axis represents the objective value of three placement algorithms. Compared to the bin-packing and random placement mechanisms, communication costs of our optimization-based placement algorithm are the lowest in all cases.

Since the optimization-based placement algorithm takes into account dependencies of multi-level applications or communication requiring frequency, it can greatly reduce the total traffic burden for the data center network.

4.3 Scenario Three: Performance Comparisons in Table

The comparison results for our approach, the bin-packing approach and the random approach are shown in Table 2. In order to highlight the effect, only the reduction rates are showed here. The (B) and (R) in sceond row are used to represent Bin-packing and Random placement approach, respectively.

Table 2 The reduction rate of new algorithm

No. of virtual machines	Four different architectures							
	Tree (B)	Tree (R)	VL2 (B)	VL2 (R)	Fat-Tree (B)	Fat-Tree (R)	BCube (B)	BCube (R)
16	0.4989	0.6833	0.4015	0.6224	0.3408	0.5825	0.7019	0.8111
32	0.5077	0.6880	0.4096	0.6254	0.3536	0.5904	0.7099	0.8163
64	0.4850	0.6740	0.3977	0.6174	0.3301	0.5738	0.6965	0.8082
128	0.4898	0.6769	0.3926	0.6139	0.3203	0.5692	0.7001	0.8107
200	0.4833	0.6736	0.3989	0.6194	0.3265	0.5727	0.6999	0.8101
256	0.4793	0.6705	0.3739	0.6027	0.3007	0.5556	0.6931	0.8059
350	0.4669	0.6537	0.3582	0.5945	0.2961	0.5457	0.6832	0.7995

We consider the impact of different network architectures as well. For testing the multiple cases, the number of virtual machines is set to 16, 32, 64, 128, 200, 256 and 350 respectively. Similar with previous scenarios, all shown results are still the average value of 1000 runs. We found that: In the Tree, VL2, Fat-Tree and BCube architectures, our optimization-based algorithm saves about 46–49, 35–40, 29–34 and 68–70 % traffic flow respectively compared with the bin-packing algorithm. However, such numerical ranges are changed to 65–68, 59–62, 54–58 and 79–81 % when comparing our algorithm with the random algorithm. Quite similar with the findings in Scenario One, system with BCube architecture can reduce 70.99 % in the best case which is comparing with bin-packing scheme. The highest value for BCube and random scheme combination is 81.63 %. The advantages for our algorithm in Fat-Tree case are reflected in 35.36 % (with bin-packing scheme) and 59.04 % (with random scheme). Although our new algorithm still wins, the performance enhancement in Fat-Tree is the most non-obvious. The results in Tree and VL2 cases are in the middle positions: For the bin-packing scheme, the greatest reduction rates are 50.77 and 40.96 %, respectively. These values in the random scheme are 68.80 and 62.54 %.

If the number of physical hosts is fixed, with the number of virtual machines increasing, reduction proportion of traffic flow gradually decreases in all cases. Especially when 350 virtual machines are generated in this scenario, the lowest value 29.61 % appears in Fat-Tree (B) column. Comparing with 256 VMs condition, the values in this row roughly decline 1–2 %.

4.4 Scenario Four: Performance Comparisons in Scatter

Our optimization-based virtual machine placement scheme can not only effectively reduce the transmission load of data center network, but also enhance the physical machine utilization and scalability. According to the reasonable deployment of algorithms, the administrator is able to adjust the number of physical machines, thereby saving the cost and energy of network equipments.

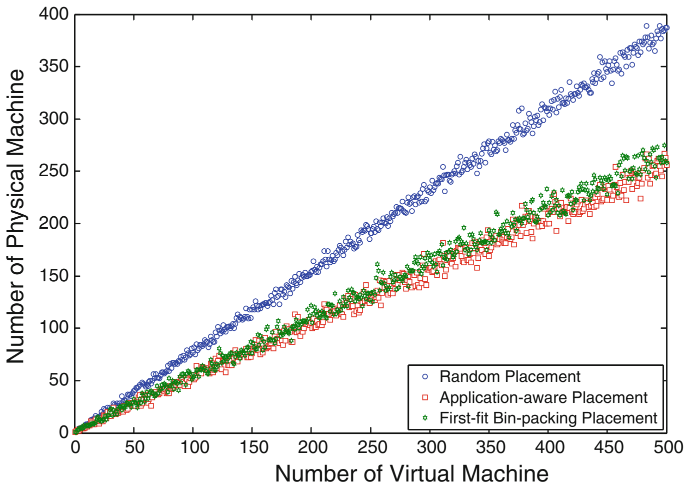

Fig. 4 Physical machines usage in three VMP algorithms

The result shows that: comparing with the traditional random placement scheme and first-fit bin-packing placement scheme (virtual machines tend to be deployed in the first physical machine that meets its load demands), our optimization-based placement scheme occupies the least number of physical machines and has the highest utilization. In the experiment process of this scenario, we still assume that the total capacity of all physical hosts is greater than the sum of load demands of all virtual machines. The optimization-based placement algorithm will assign virtual machines to the most suitable physical hosts based on the virtual machines' workload requirement (such as CPU, memory, storage space, etc.) in the VM mapping phase, which makes the final cost of the physical machines reach the global minimum.

In Fig. 4, the increase trend is basically following linear pattern. In the macroscopic perspective, random scheme take up more physical machines in most cases, and the scatter for first-fit bin-packing scheme and our scheme are rising together. The more virtual machine is generated, the more superiority of optimization-based scheme shows up. When the number of virtual machines is varying from 200 to 400, the value of physical machine is changed from 150 to 320 in random scheme. However, for our scheme, the value just roughly fluctuates in the range of 110–210. If first-fit bin-packing scheme is adopted, more physical machines are needed.

5 Conclusions and Future work

Based on the powerful convex optimization theory, we proposed an optimization-based algorithm to solve the virtual machine placement problem in large scale. Different with current existing mechanisms which only focus on one perspective (such as application awareness, server constraints, etc.), our algorithm reasonably transfer the practical problem into multiple optimization formulas. The detailed derivation procedures together with two theorems are also given. The idea discussed in this paper is quite significant in solving similar issues in virtual machine placement area.

In order to evaluate our algorithm, four widely-used data center architectures (Tree, VL2, Fat-Tree and BCube) are selected. Four different scenarios are set up in performance validation: Firstly, by using fuzzy theory, targets are unified and objective value is employed to show the advantages of optimization-based scheme. Although other architectures also obviously reduce the communication costs by using our scheme, the BCube is benefited most. Secondly, the effect in Tree, VL2 and Fat-Tree structures are deeply analyzed. From 16 to 256, multiple virtual machines are generated to calculate the objective values. Performance superiority of our scheme is illustrated clearly in this scenario. Thirdly, a table is produced to demonstrate the reduction rate. For getting a comprehensive investigation, totally 350 virtual machines are employed. In four architectures, our algorithm saves about 46–49, 35–40, 29–34 and 68–70 % traffic flow respectively compared with the bin-packing algorithm. For random algorithm, our scheme wins more. Fourthly, the physical machine utilization enhancement is tested. 500 virtual machines are used to evaluate three virtual machines placement algorithms. When the number of virtual machines is varying from 200 to 400, the number of required physical machine is changed from 150 to 320 in random scheme. However, this value just roughly fluctuates in the range of 110–210 with our scheme.

For the future work, we are planning to implement our algorithms on actual experimental environments. And it is also very important to improve the algorithm after evaluating the system performance.

Acknowledgments This work was supported in part by the Natural Science Foundation of China under Grant No. 61301081, in part by the SRFDP under Grant No. 20120009120005, in part by the MIIT of China under Grant No. 2012ZX03005003-04, in part by the Beijing Natural Science Foundation under Grant No. 4122060.

6 Appendix

6.1 Appendix A: Proof of Theorem 1

Proof The constraints of ($\boldsymbol{P}_1$) are linear and the constraint domain is convex set. In addition,

$$\frac{\partial^2 U_p}{\partial x_{pv}^2(t)} < 0 \text{ and } \frac{\partial^2 U_p}{\partial x_{pv}(t)\,\partial x_{pw}(t)} < 0 \tag{38}$$

are satisfied. Thus, the Hesse matrix of the objective function is negative definition. That means the objection function is concave but not strictly concave with respect to

$$x(t) = \left(x_{pv}(t), p \in P, v \in V\right). \tag{39}$$

According to [27], ($\boldsymbol{P}_1$) is a convex programming problem, it is easy to conclude that the optimal solution exists but not unique. Assuming that $y_1(t)$ and $y_2(t)$ are two different solutions for the total load requirements. Let α be any real number which ranges from 0 to 1. Then, since $U_p\left(y_p(t)\right) = w_p \log y_p(t)$ is the objective function, when $y_1(t) \neq y_2(t)$, clearly:

$$U_p\left(\alpha y_1(t) + (1-\alpha)\, y_2(t)\right) > \alpha U_p\left(y_1(t)\right) + (1-\alpha)\, U_p\left(y_2(t)\right). \tag{40}$$

The objection function is strictly concave with respect to $y = \left(y_p(t), p \in P\right)$, the unique optimal solution exists for total load requirements. $\square$

6.2 Appendix B: Proof of Theorem 2

Proof Based on the fact that $\boldsymbol{0} \in R$, $R \neq \varnothing$. Assuming $x_i(t) \geq 0, i = 1, 2, \ldots, n$, R is not empty bounded set. Due to the constraints of $(\boldsymbol{P}_2)$ are linear, it is ensured that R is a closed set. For the case that the objective function given by Eq. (29) is linear and continuous function, $(\boldsymbol{P_2})$ is equivalent to seek the maximum values problem upon the non-empty, bounded closed set. Therefore, the unique optimal solution exists, namely $R^* \neq \varnothing$. $\square$

6.3 Appendix C: The Expressions of Communication Distance for Different Topologies

In traditional Tree topologies, we assign n_0 as the output number of the access switch, and n_1 as the output number of the aggregation switch, then

$$Distance_{kl}^{Tree} = \begin{cases} 0 & \textit{if } k = l \\ 1 & \textit{if } \left\lfloor \frac{k-1}{n_0} \right\rfloor = \left\lfloor \frac{l-1}{n_0} \right\rfloor \\ 3 & \textit{if } \left\lfloor \frac{k-1}{n_0} \right\rfloor \neq \left\lfloor \frac{l-1}{n_0} \right\rfloor \wedge \left\lfloor \frac{k-1}{n_0 n_1} \right\rfloor = \left\lfloor \frac{l-1}{n_0 n_1} \right\rfloor \\ 5 & \textit{if } \left\lfloor \frac{k-1}{n_0 n_1} \right\rfloor \neq \left\lfloor \frac{l-1}{n_0 n_1} \right\rfloor \end{cases}. \tag{41}$$

For the VL2 topologies, in order to achieve variable load balancing, assume that all traffic from the access switch will go out through the core switch for further forwarding, setting n_0 as the output number of the access switch, then

$$Distance_{kl}^{VL2} = \begin{cases} 0 & \textit{if } k = l \\ 1 & \textit{if } \left\lfloor \frac{k-1}{n_0} \right\rfloor = \left\lfloor \frac{l-1}{n_0} \right\rfloor \\ 5 & \textit{if } \left\lfloor \frac{k-1}{n_0 n_1} \right\rfloor \neq \left\lfloor \frac{l-1}{n_0 n_1} \right\rfloor \end{cases}. \tag{42}$$

In Fat-Tree architecture, communication distance is the function of n_{port}, the total number of ports per switch, then:

$$Distance_{kl}^{Fat-Tree} = \begin{cases} 0 & if\ k=l \\ 1 & if\ \left\lfloor \frac{2(k-1)}{n_{port}} \right\rfloor = \left\lfloor \frac{2(l-1)}{n_{port}} \right\rfloor \\ 3 & if\ \left\lfloor \frac{2(k-1)}{n_{port}} \right\rfloor \neq \left\lfloor \frac{2(l-1)}{n_{port}} \right\rfloor \wedge \left\lfloor \frac{4(k-1)}{n_{port}^2} \right\rfloor = \left\lfloor \frac{4(l-1)}{n_{port}^2} \right\rfloor \\ 5 & if\ \left\lfloor \frac{4(k-1)}{n_{port}^2} \right\rfloor \neq \left\lfloor \frac{4(l-1)}{n_{port}^2} \right\rfloor \end{cases}. \tag{43}$$

In BCube framework, communication distance is the Hamming distance function of server address, then:

$$Distance_{kl}^{BCube} = \begin{cases} 0 & if\ k=l \\ 2\ hamming\left(addr\left(k\right), addr\left(l\right)\right) - 1 & if\ k \neq l \end{cases}. \tag{44}$$

References

1. Deed, C., Cragg, P.: Business Impacts of Cloud Computing. Cloud Computing Service Deployment Models: Layers and Management, pp. 274–288. IGI Global (2013)
2. Ho, S.M., Lee, H.: A thief among us: the use of finite-state machines to dissect insider threat in cloud communications. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. **3**(2), 82–98 (2012)
3. Armbrust, M., Fox, A., Griffith, R., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
4. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 68–73 (2009)
5. Shieh, A., Kandula, S., Greenberg, A., et al.: Sharing the data center network. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (2011)
6. Bari, M., Boutaba, R., Esteves, R., Granville, L., Podlesny, M., Rabbani, M., Zhang, Q., Zhani, M.: Data center network virtualization: a survey. IEEE Commun. Surv. Tutor. **15**(2), 909–928 (2013)
7. Ando, R., Takahashi, K., Suzaki, K.: Inter-domain communication protocol for real-time file access monitor of virtual machine. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. **3**(2), 120–137 (2012)
8. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. In: Proceedings of the SIGCOMM 2008 Conference on Data Communication. ACM, pp. 63–74 (2008)
9. Greenberg, A., Hamilton, J.R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D.A., Patel, P., Sengupta, S.: VL2: a scalable and flexible data center network. In: Proceedings of the SIGCOMM 2009 Conference on Data Communication. ACM, pp. 51–62 (2009)
10. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. In: Proceedings of the SIGCOMM 2008 Conference on Data Communication. ACM, pp. 75–86 (2008)
11. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C.: Bcube: a high performance, server-centric network architecture for modular data centers. In: Proceedings of the SIGCOMM 2009 Conference on Data Communication. ACM, pp. 63–74 (2009)
12. Kurp, P.: Green computing. Commun. ACM **51**(10), 11–13 (2008)
13. Baliga, J., Ayre, R.W.A., Hinton, K., Tucker, R.S.: Green cloud computing: balancing energy in processing, storage, and transport. Proc. IEEE **99**(1), 149–167 (2011)
14. Piao, J.T., Yan, J.: A network-aware virtual machine placement and migration approach in cloud computing. In Proceeding of the 9th International Conference on Grid and Cloud Computing, pp. 87–92 (2010)
15. Wang, W., Chen, H., Chen, X.: An availability-aware virtual machine placement approach for dynamic scaling of cloud applications. In: The 9th International Conference on Ubiquitous Intelligence and Computing and 9th International Conference on Autonomic and Trusted Computing, pp. 509–516 (2012)
16. VMware Capacity Planner, http://www.vmware.com/products/capacityplanner/
17. IBM WebSphere CloudBurst, http://www-01.ibm.com/software/webservers/cloudburst/

18. Novell PlateSpin Recon, http://www.novell.com/products/recon/
19. Lanamark Suite, http://www.lanamark.com/
20. Van, H.N., Tran, F.D., Menaud, J.M.: Autonomic virtual resource management for service hosting platforms. In: Software Engineering Challenges of Cloud Computing, ICSE Workshop on (2009)
21. Meng, X., Pappas, V., Zhang, L.: Improving the scalability of data center networks with traffic-aware virtual machine placement. In: Proceedings of INFOCOM (2010)
22. Bobroff, N., Kochut, A., Beaty, K.: Dynamic placement of virtual machines for managing SLA violations. Integrated Network Management. 2007. In: Proceedings of 10th IFIP/IEEE International, Symposium, pp. 119–128
23. Chaisiri, S., Lee, B.S., Niyato, D.: Optimal virtual machine placement across multiple cloud providers. In: Proceedings of IEEE Asia-Pacific Services Computing Conference, APSCC, pp. 103–110 (2009)
24. Nakada, H., Hirofuchi, T., Ogawa, H., Itoh, S.: Toward virtual machine packing optimization based on genetic algorithm. In: Proceedings of the 10th International Work Conference on Artificial Neural Networks. Springer, pp. 651–654 (2009)
25. Agrawal, S., Bose, S.K., Sundarrajan, S.: Grouping genetic algorithm for solving the server consolidation problem with conflicts. In: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, ACM pp. 1–8 (2009)
26. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
27. Chi, C.Y.: Convex Optimization for Signal Processing and Communications, to be published