Hindawi Publishing Corporation
Scientific Programming
Volume 2016, Article ID 3101658, 10 pages
http://dx.doi.org/10.1155/2016/3101658

Research Article

Optimized Virtual Machine Placement with Traffic-Aware Balancing in Data Center Networks

Tao Chen, Xiaofeng Gao, and Guihai Chen

Shanghai Key Laboratory of Scalable Computing and Systems, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

Correspondence should be addressed to Xiaofeng Gao; gao-xf@cs.sjtu.edu.cn

Received 26 February 2016; Revised 22 July 2016; Accepted 1 August 2016

Academic Editor: Ligang He

Virtualization has been an efficient method to fully utilize computing resources such as servers. The way of placing virtual machines (VMs) among a large pool of servers greatly affects the performance of data center networks (DCNs). As network resources have become a main bottleneck of the performance of DCNs, we concentrate on VM placement with Traffic-Aware Balancing to evenly utilize the links in DCNs. In this paper, we first proposed a Virtual Machine Placement Problem with Traffic-Aware Balancing (VMPPTB) and then proved it to be NP-hard and designed a Longest Processing Time Based Placement algorithm (LPTBP algorithm) to solve it. To take advantage of the communication locality, we proposed Locality-Aware Virtual Machine Placement Problem with Traffic-Aware Balancing (LVMPPTB), which is a multiobjective optimization problem of simultaneously minimizing the maximum number of VM partitions of requests and minimizing the maximum bandwidth occupancy on uplinks of Top of Rack (ToR) switches. We also proved it to be NP-hard and designed a heuristic algorithm (Least-Load First Based Placement algorithm, LLBP algorithm) to solve it. Through extensive simulations, the proposed heuristic algorithm is proven to significantly balance the bandwidth occupancy on uplinks of ToR switches, while keeping the number of VM partitions of each request small enough.

1. Introduction

As virtualization technology [1] becomes the mainstream way to multiplex various physical resources in modern cloud data centers, the effective and efficient placement of virtual machines (VMs) becomes an important issue. Mechanisms such as VMware Capacity Planner [2] and Novell PlateSpin Recon [3] consolidate VMs such that the consumption of CPU, memory, and power are optimized. Owing to the increasing deployment of communication-intensive applications like MapReduce [4], the data center network (DCN) is becoming the bottleneck of applications performance and scalability. Thus, those mechanisms without considering network resources are not feasible in cloud data centers.

Three-layer tree-like architecture is prevalently used in modern data centers [5], as shown in Figure 1. This kind of architecture, however, inherently suffers scalability issue due to the fact that links connected to the core layer switches usually transfer more traffic from lower layers. Physical machines (PMs) connected to the same Top of Rack (ToR) switch can communicate at full line speed, and the traffic between PMs connected to different ToR switches has to traverse across links of the core layer, which is often the bottleneck of data center networks (DCNs). In this paper, we placed VMs on PMs by effectively balancing traffic in the core layer of DCNs to minimize the maximum bandwidth occupancy on uplinks of the Top of Rack (ToR) switches.

The issue on scalability of DCNs has attracted great attention from academia recently. Several recent works address this issue by designing new DCN architectures, aiming at maximizing the network bisection bandwidth [6–8] and reducing the overall oversubscription ratio of the DCN. Other papers [9–11] address this issue by optimizing VM placements on PMs with different optimization goals. The bisection bandwidth can be improved; however, the available bandwidth in the core layer cannot be fully utilized. The number of highly utilized links in the core layer never exceeds 25% [12], which means a great number of links in the core layer are underutilized. Our proposed VM placement with Traffic-Aware Balancing can evenly spread traffic across links

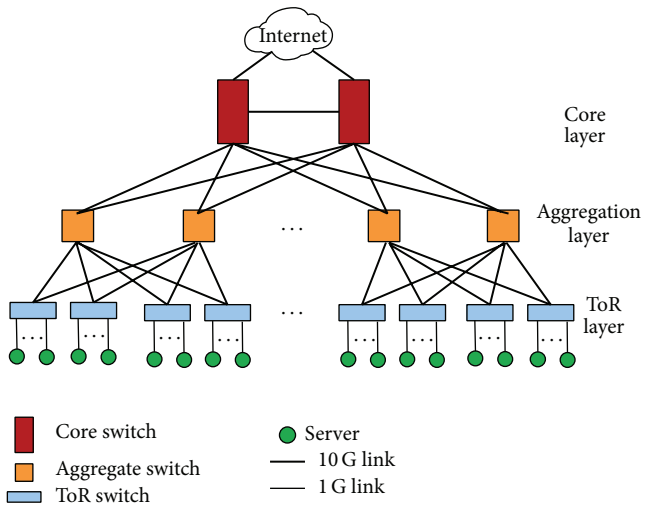

Figure 1: An example of three-layer tree-like architecture.

of the core layer to utilize the high bisection bandwidth. Since link utilizations in the core/aggregation layers are higher than that in the ToR layer and a significant fraction of the core links appear as hotspots persistently [12, 13], we only consider the traffic across the core layer of DCNs, and the traffic between VMs of the same request under the same ToR switch does not contribute any traffic towards the core layer.

We formally defined our Virtual Machine Placement Problem with Traffic Balancing (VMPPTB) as an optimization problem, which minimizes the maximum bandwidth occupancy on uplinks of each ToR switch. We proved its NP-hardness by reducing from the Multiprocessor Scheduling Problem [14] and designed a Longest Processing Time Based Placement algorithm (LPTBP algorithm) to solve it. The LPTBP algorithm provides an optimum solution to the VMPPTB problem; however, the generated placement schema tends to evenly place VMs of each request under every ToR switch. As the fact that VMs of a tenant's request only communicate with other VMs within the same request (communication locality property), we should reduce the number of VM partitions of each request at the same time (i.e., placing VMs of the same request on as few PMs as possible). We further proposed Locality-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB), which aims to minimize the maximum number of VM partitions of each request and the maximum bandwidth occupancy on uplinks of ToR switches simultaneously. We also proved it to be NP-hard by reducing from VMPPTB and designed a Least-Load First Based Placement algorithm (LLBP algorithm) to solve it.

We summarize our contribution as follows:

(i) We formally formulated the Virtual Machine Placement Problem with Traffic Balancing (VMPPTB), proved its computation complexity, and designed a Longest Processing Time Based Placement algorithm (LPTBP algorithm).

(ii) To take advantage of the communication locality property, we further proposed the Locality-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB), proved its NP-hardness, and designed a Least-Load First Based Placement algorithm (LLBP algorithm).

(iii) We designed a Greedy Based Placement algorithm (GBP algorithm) as the expected baseline, and took the LPTBP algorithm as the optimum solution. We evaluated the performance of LLBP through extensive simulations, compared with GBP algorithm and LPTBP algorithm.

The rest of this paper is organized as follows. We briefly present related work in Section 2. Problem formulation and computation complexity proofs are presented in Section 3. In Section 4, we describe the LPTBP and LLBP algorithms. We evaluate our algorithms in Section 5. Finally, we make a conclusion in Section 6.

2. Related Work

In [9], the authors addressed the network scalability issue by using traffic-aware virtual machine (VM) placement. They defined the placement problem as an optimization problem to minimize the communication costs of all the VMs, where communication cost is defined as the hops between each VM pair. They assumed that the traffic matrices between virtual machines are known in advance. The generated placement scheme can reduce the communication distance between VMs with large traffic and reduce aggregated traffic into the higher level of the data center network (DCN) architecture.

In [10], the authors proposed jointly optimizing virtual machine placement and route selecting. They strived to minimize the averaged congestion rate of every link in DCNs. Their placement strategy performs better in topologies with rich connectivity and path diversity. Although the averaged congestion rate is minimized, the traffic in DCNs may not be significantly balanced. The traffic matrix is also assumed to be known in advance.

In [11], the authors presented a Min-Cut Ratio-Aware VM Placement (MCRVMP) problem. They tried to minimize the maximum ratio of the demand and capacity across all cuts in the network, where the cut is defined as a set of links that partition the hosts into two disjoint connected components, the capacity is the sum capacity of the links, and the demand is the total traffic from either side of the hosts. In this way, each network cut may have spare capacity to absorb unpredicted traffic bursts. This work is only used in medium sized data centers, and traffic rates are assumed to be known in advance.

Knowing the traffic matrix in advance is a very strong assumption. In [15], the authors proposed to adopt the product traffic pattern model to characterize the traffic rates between VMs, where the traffic rate is defined as the product of activity levels of two communicating VMs. Similar to the objective of [9], they tended to place more active VMs into physical machines (PMs) with less communication cost. The proposed product traffic pattern is unrealistic to some degree. The generated placement scheme may result in hotspots, as VMs with higher activity levels are placed in hosts connected

to the same switch; the uplinks of that switch may become hotspots.

In [16], the authors formulated the placement problem as an optimization problem to minimize the total cost caused by network traffic and utilization of PMs. When the number of PMs is fixed, the authors proposed three different traffic cost functions to solve the optimization problem. The data center topology, however, is not taken into consideration when designing the traffic cost functions.

3. Problem Formulation

In this section, we define the VM placement problem based on tree-like architectures such as fat-tree [6] and VL2 [7]. Data centers usually leverage three-layer tree-like architectures, in which physical machines (PMs) are directly connected with Top of Rack (ToR) switches, ToR switches are connected with aggregation switches, and aggregation switches are further connected with core switches. The tree-like topology, however, is often oversubscribed, due to the fact that a higher level link has to carry traffic from several lower level links. As the higher level links are often the bottlenecks and hotspots [12], we explore to balance traffic on links of the core layer of the data center networks (DCNs).

In the VM placement problem, the traffic between VMs connected to the same ToR switch does not transfer across the core switch layer of DCNs. We only need to consider traffic on the uplinks of ToR switches, since the traffic can be dynamically load balanced among the aggregation layer and the core layer using load balancing mechanisms like Valiant Load Balancing [6]. By properly placing VMs of multiple requests, we can better evenly and effectively utilize every link of core layer of the DCNs and thus minimize the maximum bandwidth occupancy on uplinks of ToR switches.

3.1. VMPPTB Problem. We define the VM placement problem in the scenario where the DCN contains n ToR switches, represented as a set $\mathbb{T} = \{T_1, T_2, \ldots, T_n\}$. For each ToR switch, we can place up to c VMs in one PM connected with it. Let L_k be the uplink of the kth ToR switch, and let B_{L_k} be the accumulated bandwidth occupancy on uplink L_i. Suppose there are m requests $\mathbb{R} = \{R_1, R_2, \ldots, R_m\}$ from different tenants in the cloud data center, and the corresponding numbers of requested VM are $\mathbb{S} = \{s_1, s_2, \ldots, s_m \mid \forall i,\ s_i > c\}$. If $s_i \leqslant c$, the requested VMs could be placed under the same ToR switch, and when $s_i > c$, the requested VMs must be partitioned into several PMs. The bandwidth requirement of VM j of request R_i is denoted as B_{ij}. If VM j was placed under a ToR switch T_k, it would contribute B_{ij} bandwidth occupancy on uplink L_k. We should properly place all the VMs of requests under n ToR switches to minimize the maximum bandwidth occupancy on the uplinks of ToR switches with the constraint that each VM should be placed under some ToR switch and the number of VMs placed under one ToR switch should be no more than c.

Let D_{ij}^k be a binary indicator of whether VM j of request R_i is placed under ToR T_k. Consider

$$D_{ij}^k = \begin{cases} 1, & \text{VM } j \text{ of request } R_i \text{ is placed under } T_k \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The descriptions of the symbols used in this paper are summarized in Notations.

We formally define the Virtual Machine Placement Problem with Traffic Balancing (VMPPTB) as follows:

$$\min \quad \max_{k \in [1,n]} B_{L_k} \tag{2}$$

$$\text{s.t.} \quad \sum_{j=1}^{s_i} \sum_{k=1}^{n} D_{ij}^k = s_i, \quad \forall i \in \{1, 2, \ldots, m\} \tag{3}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{s_i} D_{ij}^k \leqslant c, \quad \forall k \in \{1, 2, \ldots, n\} \tag{4}$$

$$B_{L_k} = \sum_{i=1}^{m} \sum_{j=1}^{s_i} D_{ij}^k \cdot B_{ij}, \quad \forall k \in \{1, 2, \ldots, n\}. \tag{5}$$

The objective function (2) minimizes the maximum bandwidth occupancy on uplinks of ToR switches. Constraint (3) ensures that each VM of requests is placed under some ToR switch. Constraint (4) guarantees that the number of VMs under every ToR switch is not more than the capacity of one PM. Constraint (5) updates bandwidth occupancy on uplink L_k when we place a VM under ToR switch T_k.

We prove VMPPTB is a NP-hard problem.

Theorem 1. *For the Virtual Machine Placement Problem with Traffic Balancing (VMPPTB) defined above, finding its optimal solution is NP-hard.*

Proof. This can be proven by a reduction from the Multiprocessor Scheduling Problem (MSP). Given a set $\mathbb{J}$ of independent jobs and a number of processors M and given that job J_i has length I_i, what is the minimum possible time required to schedule all jobs in $\mathbb{J}$ on M processors such that none overlap? The MSP is known to be a NP-hard problem [14].

For example, we make the set $\mathbb{J}$ as a request of some tenant, job J_i as VM_j of request R_m, the length I_i of job J_i as bandwidth requirement B_{mj} of VM_j of request R_m, and M processors as n ToR switches. It turns out to be an instance of VMPPTB, except that the number of VMs placed under a ToR switch is limited to c in VMPPTB. However, we can set c extremely large such that the constraint holds in any placement schema. In this way, the MSP can be reduced to the VMPPTB. Thus, we prove the VMPPTB is a NP-hard problem. □

Since we have proven that VMPPTB is NP-hard by reducing from MSP, we can solve VMPPTB by approximation algorithms designed to solve MSP. A simple but classical algorithm called Longest Processing Time (LPT) algorithm

can achieve an upper bound of $(4/3 - 1/3M)$OPT [17]. It is feasible to design a Longest Processing Time Based Placement (LPTBP) algorithm to solve VMPPTB. We first sort bandwidth requirements of VMs of all requests in nonincreasing order. Then we place the VM with the maximum bandwidth requirement under the ToR switch, of which the uplink has minimum bandwidth occupancy. We repeat the process until VMs of all requests are placed on PMs.

3.2. LVMPPTB Problem. The LPTBP algorithm can generate an approximate optimal solution to VMPPTB; however, it cannot take the communication locality of VMs of a request into consideration (i.e., VMs of a request only communicate with other VMs within the same request). Thus, if all VMs of a request are placed under as few ToR switches as possible, it could essentially reduce the traffic forward to the core layer and further mitigate hotspots in the core layer.

Let $\mathbb{T}_{R_i} = \{T_k \mid D_{ij}^k = 1,\ \forall j \in \{1, 2, \ldots, s_i\}\}$ be a subset of $\mathbb{T}$, which contains the ToR switches under which the VMs of request R_i are placed. $|\mathbb{T}_{R_i}|$ denotes the number of VM partitions of request R_i, which should be minimized to effectively take advantage of the communication locality property. The multiobjective optimization problem named Locality-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB) aims to minimize the maximum bandwidth occupancy on the uplinks of ToR switches and minimize the maximum number of VM partitions of all the requests simultaneously, which is formally defined as follows:

$$
\begin{aligned}
&\min \quad \max_{k\in[1,n]} B_{L_k} \\
&\min \quad \max_{i\in[1,m]} \left|\mathbb{T}_{R_i}\right| \\
&\text{s.t.} \quad \sum_{j=1}^{s_i}\sum_{k=1}^{n} D_{ij}^k = s_i, \quad \forall i \in \{1, 2, \ldots, m\} \\
&\qquad \sum_{i=1}^{m}\sum_{j=1}^{s_i} D_{ij}^k \leqslant c, \quad \forall k \in \{1, 2, \ldots, n\} \\
&\qquad B_{L_k} = \sum_{i=1}^{m}\sum_{j=1}^{s_i} D_{ij}^k \cdot B_{ij}, \quad \forall k \in \{1, 2, \ldots, n\} \\
&\qquad \mathbb{T}_{R_i} = \left\{T_k \mid D_{ij}^k = 1\right\}, \quad \forall j \in \left\{1, 2, \ldots, s_i\right\}.
\end{aligned}
\tag{6}
$$

We also prove LVMPPTB is a NP-hard problem.

Theorem 2. *For the Locaility-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB), finding its optimal solution is NP-hard.*

Proof. Since we have proven that the VMPPTB is a NP-hard problem, we can prove LVMPPTB's NP-hardness by reducing it from the VMPPTB. A special instance of LVMPPTB is that there is only one request in the DCN. Therefore, the number of VM partitions of this request equals the number of VMs of this request divided by the maximum number of VMs placed under a ToR switch. It turns out to be the same with the VMPPTB. If we can find an optimal solution to the VMPPTB, we can also find an optimal solution to this special instance of LVMPPTB and vice versa. As the VMPPTB is NP-hard, the LVMPPTB's NP-hardness is proven. □

4. Algorithms

We have proven that the Virtual Machine Placement Problem with Traffic Balancing (VMPPTB) is a NP-hard problem. As the VMPPTB is the reduction from the Multiprocessor Scheduling Problem, we designed a Longest Processing Time Based Placement (LPTBP) algorithm to solve the VMPPTB, as shown in Algorithm 1.

The LPTBP algorithm aims to achieve balanced bandwidth occupancy of uplinks of all ToR switches by placing the VM with maximum bandwidth requirement under the ToR switch with minimum bandwidth occupancy every time. First, $\{B_{ij}\}$ are put into a two-dimensional array $\mathbb{V}$. $\{B_{L_k}\}$ and $\{C_k\}$ are initially $\emptyset$. For the VM with maximum bandwidth requirement, the LPTBP algorithm selects the ToR switch with minimum bandwidth occupancy, places the VM under it, and then updates $\mathbb{V}$, $\mathbb{B}$, and $\mathbb{C}$. The LPTBP algorithm repeats the process until no VMs can be placed under any ToR switches and outputs a virtual machine placement schema.

Though the LPTBP algorithm has an approximation ratio $(4/3 - 1/3M)$OPT [17], the output placement schema does not take advantage of the communication locality property. Hence, we propose a heuristic algorithm named Least-Load First Based Placement (LLBP) to solve the Locality-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB), as shown in Algorithm 2.

The LLBP algorithm places the requests in the nonincreasing order of their numbers of VMs. For each request, LLBP tries to find a minimum *empty* ToR switch set that can hold all its VMs. The ToR switch(es) in an empty ToR switch set is(are) fully available and connected to none of VMs. If the *empty* ToR switch set exists, LLBP places all VMs of the request in the nonincreasing order of bandwidth requirement under the empty ToR switch set in the least-load first way (placing the VM with maximum bandwidth requirement under the ToR switch with minimum bandwidth occupancy). If no such *empty* ToR switch set exists for a request, LLBP tries to find a minimum ToR switch set to hold its VMs and also place the VMs under the ToR switch set in the least-load first way.

LLBP first places VMs of the request with maximum number of VMs, and an *empty* ToR switch set exists to meet the communication locality property, and the number of VM partitions would be small. LLBP repeats the process until no such *empty* ToR switch set exists. At this time, for these requests with smaller numbers of VMs, LLBP tries to find some ToR switch sets to hold their VMs, which are placed under several ToR switches, and the numbers of VM partitions would rise. The worst case is that each VM of a request is placed under one ToR switch, in which the communication locality property is hard to meet. However, as the number of VMs is smaller, the number of VM partitions would be smaller and the overhead of communication between these

Input: $\{B_{ij}\}$: Set of bandwidth requirements of VMs of all requests;
$\{B_{L_k}\}$: Set of cumulative bandwidth occupancy of the uplinks of all ToR switches;
$\{C_k\}$: Set of cumulative numbers of VMs under ToR switches
B_{L_c}: Maximum bandwidth capacity of a uplink of ToR switch;
c: Maximum number of VMs under a ToR switch.
Output: Virtual Machine Placement Schema
(1) $\mathbb{V} \leftarrow \{B_{ij}\}, i \in \{1, 2, \ldots, m\}, j \in \{1, 2, \ldots, s_i\}$
(2) $\mathbb{B} \leftarrow \{B_{L_k}\}, k \in \{1, 2, \ldots, n\}$
(3) $\mathbb{C} \leftarrow \{C_k\}, k \in \{1, 2, \ldots, n\}$
(4) $\mathbb{B} = \emptyset$
(5) $\mathbb{C} = \emptyset$
(6) **for** $v \leftarrow 1$ to $\sum_{i=1}^{m} s_i$ **do**
(7) $u \leftarrow \arg\min_k B[k]$
(8) $(p, q) \leftarrow \arg\max_{i,j} V[i][j]$
(9) **while** $(B[u] + V[p][q] \leqslant B_{L_c}$ && $C[u] \leqslant c)$ **do**
(10) place VM $q + 1$ of request R_{p+1} under ToR switch T_{u+1}
(11) $B[u] \leftarrow B[u] + V[p][q]$
(12) $C[u] \leftarrow C[u] + 1$
(13) $0 \leftarrow V[p][q]$
(14) **end while**
(15) **end for**
(16) **return** Virtual Machine Placement Schema

Algorithm 1: Longest Processing Time Based Placement (LPTBP).

Input: $\mathbb{S} = \{s_i\}$: Set of numbers of VMs of requests
$\{B_{ij}\}$: Set of bandwidth requirements of VMs of all requests;
$\{B_{L_k}\}$: Set of cumulative bandwidth occupancy of the uplinks of all ToR switches;
$\{C_k\}$: Set of cumulative numbers of VMs under ToR switches
B_{L_c}: Maximum bandwidth capacity of a uplink of ToR switch;
c: Maximum number of VMs under a ToR switch.
Output: Virtual Machine Placement Schema
(1) $\mathbb{S} \leftarrow \{s_i\}, i \in \{1, 2, \ldots, m\}$
(2) $\mathbb{V} \leftarrow \{B_{ij}\}, i \in \{1, 2, \ldots, m\}, j \in \{1, 2, \ldots, s_i\}$
(3) $\mathbb{B} \leftarrow \{B_{L_k}\}, k \in \{1, 2, \ldots, n\}$
(4) $\mathbb{C} \leftarrow \{C_k\}, k \in \{1, 2, \ldots, n\}$
(5) $\mathbb{B} = \emptyset$
(6) $\mathbb{C} = \emptyset$
(7) **for** $i \leftarrow 1$ to m **do**
(8) $u \leftarrow \arg\max_i S[i]$
(9) find a minimum *empty* ToR switch set $\mathbb{T}_{R_{u+1}}$ to hold all VMs of request R_{u+1}, $|\mathbb{T}_{R_{u+1}}| = \lceil s_{u+1}/c \rceil$
(10) **if** $\mathbb{T}_{R_{u+1}} = \emptyset$ **then**
(11) find a minimum ToR switch set $\mathbb{T}'_{R_{u+1}}$ to hold all VMs of request R_{u+1}
(12) **end if**
(13) place all VMs of request R_{u+1} in the non-increasing order of bandwidth requirement under $\mathbb{T}_{R_{u+1}}$ or $\mathbb{T}'_{R_{u+1}}$ in the least-load first way under constraints B_{L_k}, c
(14) update $\mathbb{B}, \mathbb{C}$ by $V[u][0], V[u][1], \ldots, V[u][s_{u+1} - 1], s_{u+1}$
(15) **end for**

Algorithm 2: Least-Load First Based Placement (LLBP).

VMs is smaller. The VMs of all requests are placed in a least-load first way, and the bandwidth occupancy on uplinks of ToR switches can be balanced significantly. Therefore, LLBP achieves a better trade-off between the number of VM partitions of requests and bandwidth occupancy of uplinks of the ToR switches.

We also design a Greedy Based Placement (GBP) algorithm, as shown in Algorithm 3. The GBP algorithm places VMs of all requests sequentially under ToR switches in the ToR switch order, which means that each VM of a request is placed as close as possible to other VMs of the same request.

Input: $\{B_{ij}\}$: Set of bandwidth requirements of VMs of all requests;
$\{B_{L_k}\}$: Set of cumulative bandwidth occupancy of the uplinks of all ToR switches;
$\{C_k\}$: Set of cumulative numbers of VMs under ToR switches
B_{L_c}: Maximum bandwidth capacity of a uplink of ToR switch;
c: Maximum number of VMs under a ToR switch.
Output: Virtual Machine Placement Schema
(1) $\mathbb{V} \leftarrow \{B_{ij}\}, i \in \{1, 2, \ldots, m\}, j \in \{1, 2, \ldots, s_i\}$
(2) $\mathbb{B} \leftarrow \{B_{L_k}\}, k \in \{1, 2, \ldots, n\}$
(3) $\mathbb{C} \leftarrow \{C_k\}, k \in \{1, 2, \ldots, n\}$
(4) $\mathbb{B} = \emptyset$
(5) $\mathbb{C} = \emptyset$
(6) **for** $i \leftarrow 1$ to m **do**
(7) **for** $j \leftarrow 1$ to s_i **do**
(8) **for** $k \leftarrow 1$ to n **do**
(9) **while** $(B[k-1] + V[i-1][j-1] \leqslant B_{L_c}$ && $C[k-1] \leqslant c)$ **do**
(10) place VM j of request R_i under ToR switches in the ToR switch order
(11) **end while**
(12) **end for**
(13) **end for**
(14) **end for**

Algorithm 3: Greedy Based Placement (GBP).

We illustrate the three algorithms by an example, as shown in Figure 2. There are 3 requests, R_1, R_2, and R_3, which are placed under 5 ToR switches, T_1, T_2, T_3, T_4, and T_5. A ToR switch can be connected with up to three VMs. The numbers of VMs of 3 requests are 4, 5, and 6, respectively. The sets of bandwidth requirements of VMs of requests are $\{25, 125, 225, 325\}$, $\{50, 150, 250, 350, 450\}$, and $\{100, 200, 300, 400, 500, 600\}$, respectively. The results of the three algorithms are shown in Figures 2(a)–2(c). The output placement schema of LTPBP algorithm achieves the best balance on the bandwidth occupancy of uplinks of ToR switches (775~825). However, the numbers of VM partitions of requests are all more than 3. The output placement schema of LLFBP algorithm achieves a trade-off between bandwidth occupancy (650~1100) and locality (the numbers of VM partitions of requests are all less than 2). Although the GBP algorithm guarantees the locality (the numbers of VM partitions are all less than 2), it is the worst in balancing the bandwidth occupancy (375~1500), which varies significantly. Let α be the ratio of maximum bandwidth occupancy over minimum bandwidth occupancy. As shown in Figure 2, α_{GBP} is 4, α_{LPTBP} is about 1.06, and α_{LLBP} is about 1.76. We originally set B_{L_c} infinity as a default. If we set B_{L_c} a value, the output placement schema might be changed. If $B_{L_c} \geqslant 825$ in LPTBP, the output VM placement schema would be not changed. When $B_{L_c} < 825$, more ToR switches are needed. If $B_{L_c} = 1000$ in LLBP, the new output VM placement schema is shown in Figure 2(d), in which the schema swaps a VM of R_3 for a VM of R_1. Though R_3 has more VM partitions, the bandwidth occupancy of uplinks is more balanced.

5. Evaluation

In this section, we evaluate the performance of the proposed heuristic algorithm (Least-Load First Based Placement, LLBP algorithm) by extensive simulations with different settings. We implement a Greedy Based Placement (GBP) algorithm and make it as the expected baseline of the algorithm performance. The Longest Processing Time Based Placement (LPTBP) algorithm can be taken as the optimal solution of the algorithm performance. We compare the performance of LLBP heuristic algorithm against the GBP and LPTBP algorithms. We first present the simulation settings and then show the evaluation results and corresponding analyses.

5.1. Simulation Settings. In our simulations, we set the five parameters as follows: the number of ToR switches nToR, the number of VMs that can be placed under each ToR switch nCap, the number of requests nReq, the number of VMs of each request nVMReq, and bandwidth requirements of each requested VMs nBnd. The scale of DCNs is denoted as nToR and nCap. In the simulations, we set $(1000, 80)$, $(1500, 60)$, $(2000, 40)$, and $(3000, 30)$. nVMReq is drawn from a uniform distribution between 80 and 120. nBnd is drawn from a normal distribution with a mean of μ and a variance of σ, where μ is drawn from a uniform distribution between 40 and 100 and σ is drawn from a uniform distribution between 0 and 40. nReq is determined by other parameters. We assume that all available VM slots are occupied by the VMs of requests.

5.2. Simulation Results. We run the GBP algorithm, LPTBP algorithm, and LLBP heuristic algorithm on the same randomly generated dataset under each scale of DCN for 100 rounds. We record the minimum bandwidth occupancy and maximum bandwidth occupancy on the uplinks of all the ToR switches for each round and get the average values shown in Figure 3. It is clear that the performance of the GBP algorithm is the worst, the LPTBP algorithm is the best,

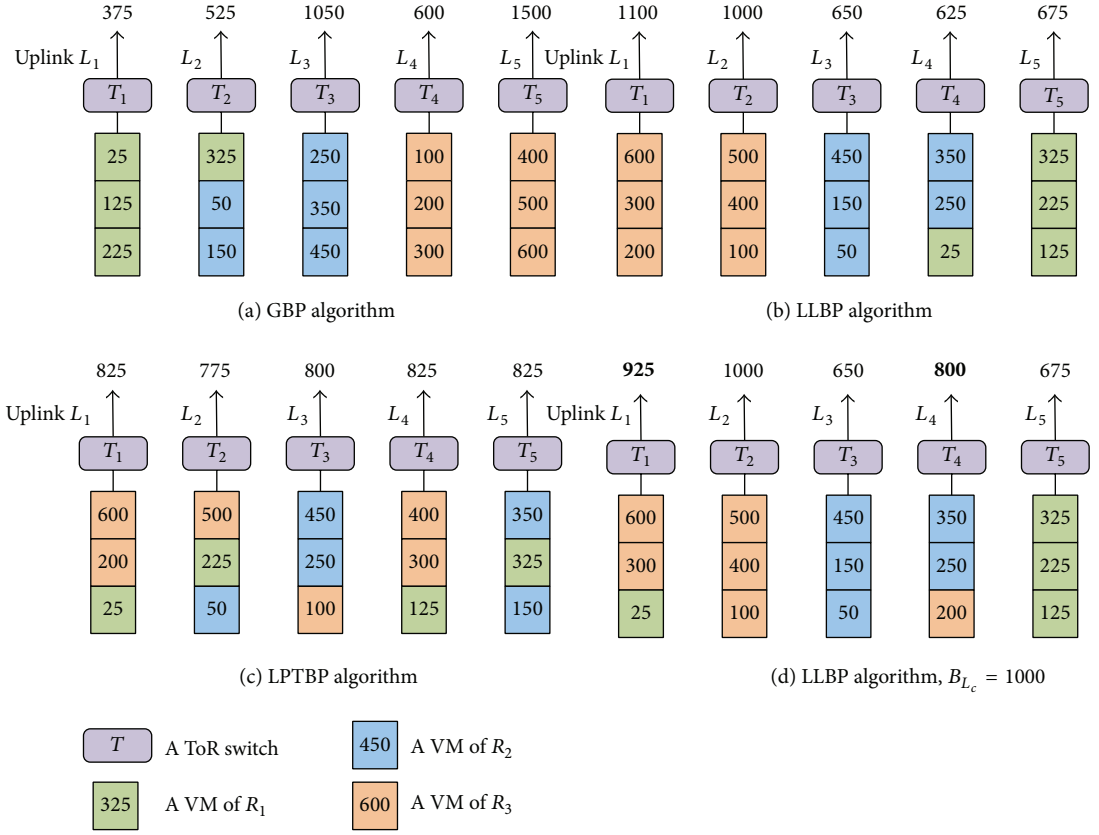

Figure 2: An example of comparison on the GBP, LLBP, and LPTBP algorithms.

and the LLBP algorithm is in the middle. According to the simulation results, α_{GBP} is larger than 3, α_{LPTBP} is nearly 1, and α_{LLFBP} is about 1.5. α values are close to the results of example in Section 4.

Although the GBP algorithm places as many as possible VMs of the same request under the same ToR switch, fully taking advantage of the communication locality property, the bandwidth occupancy on uplinks of all the ToR switches varies significantly. The LPTBP algorithm can spread the traffic of VMs evenly across all the uplinks of the ToR switches; however, it distributes the VMs of a request under several ToR switches. The proposed LLBP algorithm simultaneously balances the bandwidth occupancy on uplinks of all the ToR switches and takes advantage of the communication locality property. From the simulation results and analyses, we can conclude that LLBP algorithm performs effectively under different scales of DCNs.

6. Further Optimization

6.1. Locality. The classification of the topologies of data center networks (DCNs) falls under switch-centric, server-centric, and other topologies, according to their structural features. Switch-centric topologies consist of tree-like [6, 7, 18, 19], flat [20], and optical topologies [21, 22]. Server-centric topologies are divided into topologies designed for mega data centers [23, 24] and modular data centers (MDCs) [25, 26]. Other topologies include unstructured [27, 28] and wireless topologies [29, 30]. In DCNs, the locality can be defined as different levels. For example, in the fat-tree [6] in Figure 4, we can logically categorize locality into ToR level (S_0 and S_1), pod level (S_0 and S_2), and tree level (S_0 and S_4). Locality is different physically, such as server level, rack level, and row level. We can set different values for various locality levels in optimization.

6.2. Different Optimization Goal. In Section 3, we proposed Locality-Aware Virtual Machine Placement Problem with Traffic-Aware Balancing (LVMPPTB), which is a multiobjective optimization problem of simultaneously minimizing the maximum number of VM partitions of requests and minimizing the maximum bandwidth occupancy on uplinks of ToR switches. We also can optimize one objective while the other objective is fixed. For example, we can restrict the maximum bandwidth on uplink of a ToR switch to a fixed value to optimize the number of VM partitions of requests.

6.3. Online Balancing. VM placement problem occurs in the scenario of running applications from the very beginning, which is addressed by an offline algorithm. Once the VMs of applications are running, various problems could occur at different time, while we should use an online algorithm to perform VM migration for load balancing and fault tolerance.

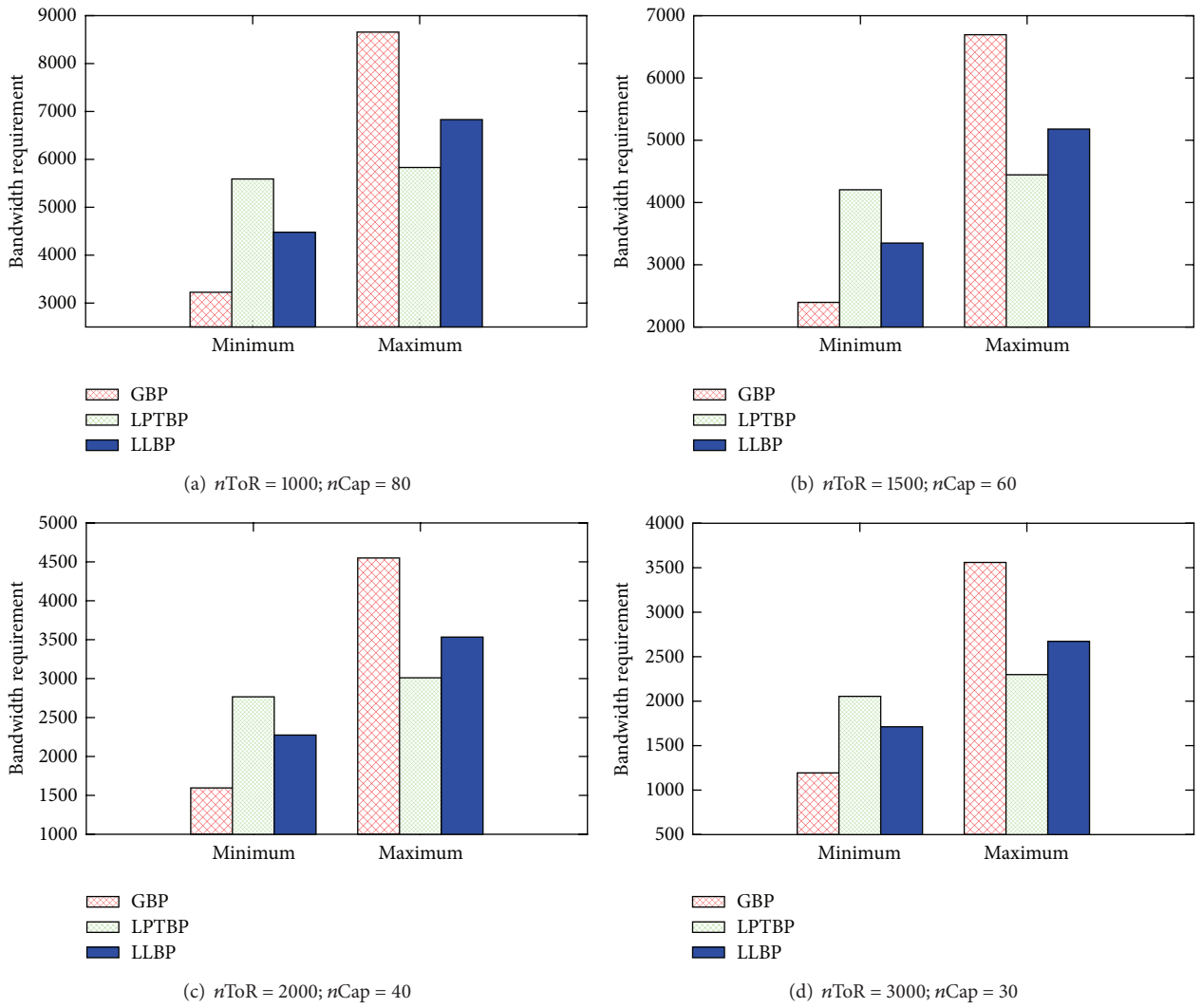

Figure 3: Minimum and maximum bandwidth requirement of different algorithms under different simulation settings.

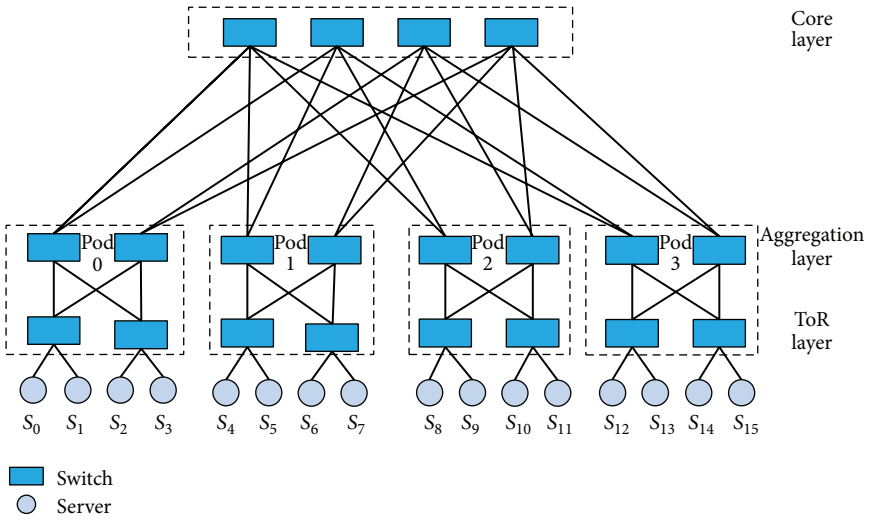

Figure 4: An example of fat-tree architecture.

We also can consider the correlations between several VMs to optimize the VM migration [31].

6.4. Fault Tolerance. In LPTBP and LLBP, we assumed that all requests of tenants were running normally. If the event of requests being lost occurs, we need to design a fault tolerant mechanism to retrieve the lost request. Timeout retransmission may be a way of solving the problem.

7. Conclusion

In this paper, we formulated Virtual Machine Placement Problem with Traffic Balancing (VMPPTB) to balance traffic in data centers. We proved its NP-hardness and designed a Longest Processing Time Based Placement algorithm. To take advantage of the communication locality property, we proposed Locality-Aware Virtual Machine Placement Problem with Traffic Balancing (LVMPPTB) which simultaneously minimizes the maximum number of VM partitions of each request and minimizes the maximum bandwidth occupancy on uplinks of ToRs. We proved its NP-hardness and designed a Least-Load First Based Placement heuristic algorithm. We conducted extensive simulations to evaluate the performance of algorithms.

Notations

$\mathbb{T} = \{T_k\}$: Set of ToR switches
c: Maximum number of VMs under a ToR switch
C_k: Number of VMs under ToR switch T_k
L_k: Uplink of ToR switch T_k
B_{L_k}: Accumulated bandwidth occupancy on uplink L_k
B_{L_c}: Maximum bandwidth capacity of uplink
$\mathbb{R} = \{R_i\}$: Set of requests from tenants
$\mathbb{S} = \{s_i\}$: Set of number of requested VMs of each request
B_{ij}: Bandwidth requirement of VM j of request R_i
D_{ij}^k: Indicator of whether VM j of request R_i is under T_k
$\mathbb{T}_{R_i}$: Set of ToR switches connected with VMs of request R_i.

Disclosure

The opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies or the government.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work has been supported in part by the China 973 Project (2014CB340303), China NSF Projects (nos. 61472252 and 61133006), the Opening Project of Key Lab of Information Network Security of Ministry of Public Security (the Third Research Institute of Ministry of Public Security) (Grant no. C15602), the Opening Project of Baidu (Grant no. 181515P005267), and the Open Project Program of Shanghai Key Laboratory of Data Science (no. 201609060001). The authors also would like to thank Tao Liang for his contributions on the early versions of this paper.

References

[1] P. Barham, B. Dragovic, K. Fraser et al., "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, vol. 37, no. 5, pp. 164–177, ACM, Lake George, NY, USA, October 2003.

[2] VMware Capacity Planer, http://www.vmware.com/products/capacity-planner.

[3] Novell PlateSpin Recon, http://www.novell.com/products/recon.

[4] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[5] Cisco Data Center Infrastructure 2.5, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/DC_Infra2_5/DCI_SRND_2_5a_book.html.

[6] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[7] A. Greenberg, J. R. Hamilton, N. Jain et al., "Vl2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

[8] R. Niranjan Mysore, A. Pamboris, N. Farrington et al., "Portland: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 39–50, 2009.

[9] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE INFOCOM*, pp. 1–9, IEEE, San Diego, Calif, USA, March 2010.

[10] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 2876–2880, Orlando, Fla, USA, March 2012.

[11] O. Biran, A. Corradi, M. Fanelli et al., "A stable network-aware VM placement for cloud systems," in *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid '12)*, pp. 498–506, IEEE, Ottawa, Canada, May 2012.

[12] 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 Internet Measurement (IMC '10)*, pp. 267–280, ACM, Melbourne, Australia, November 2010.

[13] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

[14] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1990.

[15] K. You, B. Tang, and F. Ding, "Near-optimal virtual machine placement with product traffic pattern in data centers," in *Proceedings of the 2013 IEEE International Conference on Communications (ICC '13)*, pp. 3705–3709, IEEE, Budapest, Hungary, June 2013.

[16] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: towards traffic aware virtual machine placement in data centers," in *Proceedings of the 33rd IEEE Conference on Computer Communications (INFOCOM '14)*, pp. 1842–1850, Toronto, Canada, May 2014.

[17] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[18] B. Heller, S. Seetharaman, P. Mahadevan et al., "Elastictree: saving energy in data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI '10)*, p. 17, USENIX Association, 2010.

[19] M. Walraed-Sullivan, A. Vahdat, and K. Marzullo, "Aspen trees: balancing data center fault tolerance, scalability and cost," in *Proceedings of the 9th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*, pp. 85–96, December 2013.

[20] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 338–347, 2010.

[21] K. Chen, A. Singla, A. Singh et al., "OSA: an optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Transactions on Networking*, vol. 22, no. 2, pp. 498–511, 2014.

[22] K. Chen, X. Wen, X. Ma et al., "WaveCube: a scalable, fault-tolerant, high-performance optical data center architecture," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '15)*, pp. 1–9, Hong Kong, April-May 2015.

[23] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 75–86, 2008.

[24] D. Li and J. Wu, "On the design and analysis of data center network architectures for interconnecting dual-port servers," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM '14)*, pp. 1851–1859, Toronto, Canada, April-May 2014.

[25] C. Guo, G. Lu, D. Li et al., "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63–74, 2009.

[26] D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," in *Proceedings of the 2011 19th IEEE International Conference on Network Protocols (ICNP '11)*, pp. 256–265, IEEE, Vancouver, Canada, October 2011.

[27] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: networking data centers randomly," in *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI '12)*, pp. 1–14, San Jose, Calif, USA, April 2012.

[28] A. R. Curtis, T. Carpenter, M. Elsheikh, A. López-Ortiz, and S. Keshav, "REWIRE: an optimization-based framework for unstructured data center network design," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 1116–1124, Orlando, Fla, USA, March 2012.

[29] J.-Y. Shin, E. G. Sirer, H. Weatherspoon, and D. Kirovski, "On the feasibility of completely wireless datacenters," in *Proceedings of the 8th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '12)*, pp. 3–14, Austin, Tex, USA, October 2012.

[30] X. Zhou, Z. Zhang, Y. Zhu et al., "Mirror mirror on the ceiling: flexible wireless links for data centers," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 443–454, 2012.

[31] W. Wei, X. Wei, T. Chen, X. Gao, and G. Chen, "Dynamic correlative VM placement for quality-assured cloud service," in *Proceedings of the IEEE International Conference on Communications (ICC '13)*, pp. 2573–2577, IEEE, Budapest, Hungary, June 2013.