Joint Optimization of Server and Network Resource Utilization in Cloud Data Centers

Biyu Zhou*†, Jie Wu‡, Lin Wang§, Fa Zhang¶, and Zhiyong Liu*‖
*Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, CAS
†University of Chinese Academy of Sciences, Beijing, China
‡ Center for Networked Computing, Temple University, USA §Technische Universität Darmstadt, Germany
¶Key Lab of Intelligent Information Processing, ICT, CAS
‖State Key Laboratory for Computer Architecture, ICT, CAS
{zhoubiyu, zhangfa, zyliu}@ict.ac.cn, jiewu@temple.edu, wang@tk.tu-darmstadt.de

***Abstract*—Virtual machine placement is a key component of cloud resource management, which may affect network bandwidth allocation. In this paper, we revisit the virtual machine placement problem in cloud data centers and aim to maximize the overall resource utilization in multiple dimensions, while ensuring that the resource constraints on both the server such as CPU capacity and the network such as bandwidth are not violated. We model the bandwidth-guaranteed virtual machine placement problem and prove its NP-hardness, and design offline and online algorithms to solve the problem. We first consider the offline version and develop approximation algorithms with bounded performance ratios for both the homogeneous and the heterogeneous cases. Then, for the online version, we propose simple and efficient heuristics based on the insights from the offline algorithm design. Comprehensive experimental results verify that the overall resource utilization can be significantly improved by applying our proposals.**

I. Introduction

Virtualization has already become a widely adopted technology for resource sharing in cloud computing. The ability to virtualize physical resources, such as servers and networks, provides immense benefits in terms of flexibility, reliability, and scalability [1]. In a typical cloud data center, cloud tenants submit their resource requests in the form of virtual machines (VMs) to the cloud provider. The cloud provider then decides the allocation of the physical resources to each of the requested virtual machines. By consolidating the VMs from different tenants to as less servers as possible, resource utilization can be improved and consequently, the operational cost can be reduced, while still satisfying the given tenant requests.

Although VM placement plays a vital role in cloud resource management [2, 3], achieving efficient resource allocation through VM placement is not trivial and remains as a critical issue. There has been a great number of proposals for VM placement in data centers, see [4, 5] for recent published surveys. Most of them are focused on only single resource dimensions [2, 6, 7], which makes them impractical when it comes to real scenarios as VM placement usually intertwines with multiple types of resources including CPU, memory, and network bandwidth. Recently, there are also some papers considering multi-dimensional resource allocation [8–15]. However, they still share some common drawbacks: (1) The constraint programming based models require prolonged search time in finding a reasonably good solution, while the bin-packing based heuristics cannot provide any performance guarantee. (2) While incorporating multiple resources in the server, they fail to consider the constraints on network resources. In fact, data center networks in use always suffer unpredictable performance due to oversubscription and a lack of application level bandwidth guarantee [16]. Although a lot of attention have been drawn to solve this problem, such as providing bandwidth reservations for VMs and fair bandwidth sharing among tenants, there still remains a research gap in combining VM placement and bandwidth guarantee.

In this paper, we revisit the VM placement problem and consider jointly optimizing the resource utilizations of both the server and the network. More specifically, we aim to achieve the best overall resource utilization while **providing bandwidth guarantee on the network**. We adopt the dual vector bin-packing model [1], where a VM is abstracted as a demand vector and each element in the vector represents the resource requirement in the corresponding resource dimension. To better characterize the resource demands on multiple dimensions, we introduce a new definition for the "size" of a VM. Our goal is to select VMs and to find the placement strategy for them such that the highest overall utilization for all the resources can be achieved, i.e., the total size of the accommodated VMs is maximized.

We conduct the investigation into the problem following two steps. We first focus on the offline version where all the VM demands are known *a priori*. As the network bandwidth is shared by multiple servers, we study the influence of different bandwidth allocations to our objective. We further classify the problem into two cases: homogeneous and heterogeneous. The former assumes that all the servers and network elements are identical and the latter tackles the general case where we have heterogeneous hardware in the data center. We then study the online version, where the arrival of VMs is revealed gradually and no *a priori* knowledge on the VM demands is assumed.

Compared with many previous studies [8–14] on VM

This work is supported partially by the National Natural Science Foundation of China (NSFC) Major International Collaboration Project 61520106005, NSFC Project for Innovation Groups 61521092, and the German Research Foundation (DFG) Collaborative Research Center (CRC) 1053 – MAKI.

placement of multi-dimensional resources, the present work is characterized by the joint optimization of the server resource and the bandwidth resource that is shared among multiple servers in the case of data center where the higher layers of the network are highly oversubscribed. Furthermore, both offline solutions with bounded performance ratios and online solutions which are simple and efficient are presented. The contributions are summarized as follows: (1) We define and model a VM placement problem with bandwidth guarantee for joint maximization of the server and network resource utilization. Furthermore, we prove that the problem is NP-hard in general. (2) For the offline version in which all the VMs are known *a priori*, we classify the problem into homogeneous and heterogeneous cases depending on whether the hardware is identical or not. We develop bounded approximation algorithms for both cases. (3) For the online version in which VMs are revealed over time and decisions have to be made on the fly, we present two efficient heuristics for the online VM placement problem by exploring the insights from our offline algorithms. (4) We evaluate our algorithms with comprehensive simulations, and experimental results verify that our proposal can achieve good overall resource utilization.

The remainder of this paper is organized as follows. Section II provides the problem statement and the NP-hardness proof. Section III presents the proposed bounded approximate solutions for the offline setting. Section IV introduces two efficient heuristics for the online setting. Section V shows the experimental results and Section VI concludes the paper.

II. PROBLEM STATEMENT AND ANALYSIS

In this section, we first describe the scenario and introduce some notations. Then, we provide a formal definition for the bandwidth-guaranteed VM placement problem. Finally, we carry out complexity analysis on the problem.

A. Problem Description

Assume that there is a set of m servers $\mathcal{S} = \{s_i | 1 \leq i \leq m\}$. Each server is equipped with a certain amount of resources. For the simplicity of expression, we use only CPU cycles to represent the resources considered on the server. However, our model can be easily extended to incorporate multiple resources. Each $s_i \in \mathcal{S}$ is associated with a positive value C_i to represent its capacity and we denote by $C = \{C_i | 1 \leq i \leq m\}$ the capacity set. All the servers are connected by an undirected network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{P} \cup \mathcal{S}$, $\mathcal{P}$ is the set of switches, and $\mathcal{E}$ is the set of links. The network considered in this paper is a commonly adopted three-tier architecture in data centers [17], where $\mathcal{G}$ follows a multi-rooted tree-based network topology composed of three layers of network switches, namely access, aggregate, and core layers. Each server in $\mathcal{S}$ is connected directly to one of the access layer switches. The aggregate layer switches provide connectivity for all the access layer switches, while they are interconnected by the core layer switches. Apart from that, the core layer switches are also responsible for connecting the data center to the Internet. As stated in [18], the higher layers of the three-tier data center network architecture are usually highly oversubscribed. Thus, the uplink bandwidth capacity of the core switch is often the bottleneck of the entire network.

Real data from 17K VMs in a data warehouse hosted by IBM Global Service show that there exist two types of traffic patterns for VMs: the external traffic from each VM to the Internet and the internal traffic among VMs. In some cases, VMs only send (receive) data to (from) the Internet without any internal communication [7]. We will specifically focus on the external traffic as it has been overlooked in the literature. Formally, each VM r_i is associated with a resource demand vector $\langle c_i, b_i \rangle$, where c_i represents the amount of resources required on the server (i.e., the number of CPU cycles in our case), and b_i represents the amount of required network resources (i.e., the minimum amount of bandwidth required to guarantee its connectivity to the Internet). As we are considering an oversubscribed network where the bandwidth of the core layer switches is the bottleneck of the entire network, the above VM model that captures the bandwidth demand on the cores switches from VMs will address the most critical challenge in bandwidth-guaranteed VM placement. We are also aware that incorporating network constraints in VM placement is already complicated so the above model servers as a starting point to investigate the influence of the VM placement to the overall resource utilization of both the server and the network. Nonetheless, we keep in mind to include both traffic patterns in the VM model in our follow-up work.

Given a set of VMs, $\mathcal{R} = \{r_i | 1 \leq i \leq n\}$, the goal of the *Bandwidth-Guaranteed Virtual Machine Placement problem (**BG-VMP**)* is to pack VMs into the physical servers such that the overall resource utilization is maximized while no constraints on either the server or the network are violated. To better characterize the overall resource utilization, a simple metric *size* is introduced in this paper.

Definition 1. *The size of a VM r_i is defined as the product of the resource demands on all resource dimensions, which in our case is given by $c_i b_i$.*

Note that the metric *size* can portray the level of load along two dimensions in a unified manner. The intuition is that the *size* will enlarge with the utilization increase of any resource dimensions. A similar metric is introduced in [19] to capture the combined CPU-network-memory load of a physical server. Based on the above definition, we formally define

Definition 2. ***(BG-VMP)** We are given m servers that are connected by a three-tier datacenter network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of VM requests $\mathcal{R} = \{r_i | 1 \leq i \leq n\}$, each of which is associated with a resource vector $\langle c_i, b_i \rangle$. We seek to find a subset $\mathcal{A}$ of the VM requests, i.e., $\mathcal{A} \subseteq \mathcal{R}$, and a partition $\mathcal{A}_1, \cdots, \mathcal{A}_m$ of $\mathcal{A}$, such that the CPU capacity of each server and the bandwidth capacity of each link in the network are not violated, if we place the m components of the partition to the m servers. Our goal is to maximize the total size of the VMs in the selected subset, i.e., $\sum_{i \in [1,m]} \sum_{r_j \in \mathcal{A}_i} c_j b_j$.*

B. Complexity Analysis

Theorem 1. *BG-VMP is NP-hard.*

Proof: We conduct the proof by building a polynomial-time reduction from the dual bin-packing problem which is known to be NP-hard [20]. The optimization version of the dual bin-packing problem is described as follows.

Assume we are given a set of bins $\mathcal{S} = \{s_i | 1 \leq i \leq m\}$, each bin is associated with a positive value C_i; there are a set of items $\mathcal{R} = \{r_i | 1 \leq i \leq n\}$, each of which is associated with a positive value c_i. The problem is to find a subset of the items and a partition $\mathcal{A}_1, \cdots, \mathcal{A}_m$ of this subset into the m bins, such that the sum of the items in partition $\mathcal{A}_i$ (i.e., $\sum_{r_j \in \mathcal{A}_i} c_j$) is at most C_i. The objective is to maximize the total size of the packed items in the subsect, i.e., $\max \sum_{i \in [1,m]} \sum_{r_j \in \mathcal{A}_i} c_j$.

Given any instance of the dual bin-packing problem, we can always construct from it an instance of the BG-VMP problem. We construct a set of servers from the set of bins $\mathcal{S} = \{s_i | 1 \leq i \leq m\}$, and use the value C_i to represent the CPU capacity of each server s_i. Then, we construct a three-tier network $\mathcal{G}$ that connects all the servers at the leaf. Both the bandwidth capacity of each link in the network and the uplink bandwidth of each core layer switch are assumed to be infinite. We construct a set of VMs from the set of items $\mathcal{R} = \{r_i | 1 \leq i \leq n\}$, and use the value c_i to represent the CPU demand of each VM r_i. The bandwidth demand of each VM is assumed to be one unit. It can be verified that as long as we find the optimal placement for an instance of the BG-VMP problem that maximizes the size of the packed VMs to all the servers, we can always obtain accordingly the optimal solution for the dual bin-packing instance by mapping the bins to the servers and the items to the VMs. Consequently, the BG-VMP is at least as hard as the NP-hard dual bin-packing. ■

III. Offline Algorithms

Theorem 1 reveals that obtaining the optimal solution of the BG-VMP problem with a large input instance is nearly impossible due to its high complexity. As a result, we will focus on how to obtain efficient approximate solutions in this paper. We observe that the main difficulty in solving the BG-VMP problem comes from two aspects: i) The total uplink bandwidth resource connecting to the Internet is shared among all the servers. ii) The overall resource utilization of a cloud datacenter is conditioned by the utilizations of both the server CPU resource and the network link bandwidth resource. In what follows, we first consider different allocation strategies for shared uplink bandwidth resource among all the servers, and analyze the influence of each of these strategies to the overall resource utilization. We will then deal with the two-dimensional resource allocation problem in the next sections. Before going into details, we first define that a bandwidth allocation is *feasible* if all the link bandwidth capacity constraints are not violated.

Definition 3. *(Total Resource Capacity) For a given set of servers $\mathcal{S} = \{s_i | 1 \leq i \leq m\}$ (each server is associated with a positive value C_i representing the associated CPU capacity) and a given feasible allocation of uplink bandwidth capacity for each server, $\mathcal{B} = \{B_i | 1 \leq i \leq m\}$, the Total Resource Capacity can be computed by $\sum_{i \in [1,m]} C_i B_i$.*

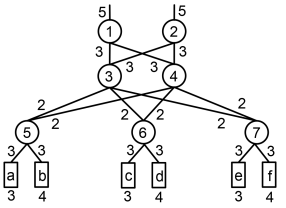

Fig. 1. An example of three-tier network architecture with 7 switches and 6 servers. Each link is labelled with a value representing the bandwidth capacity; Each server is also labelled with a value representing the server CPU capacity.

We further define that a bandwidth allocation is *optimal* if the total resource capacity cannot be increased by applying any adjustment on the allocation. It is then straightforward that the total resource capacity under an optimal bandwidth allocation is a nature upper bound for the BG-VMP problem.

An example is illustrated in Figure 1 to show the relationship between bandwidth allocation and the maximum total resource capacity. We can see that the maximum bandwidth that can be reserved for each server to the Internet is 3 units. However, the server a and the server b share bandwidth capacities in multiple links. If one reserves 3 units of bandwidth for the server a, then the server b can get at most 1 unit of bandwidth.The same situation occurs in server c and server d. Server f gets the remaining 2 units of bandwidth. The maximum total size of accommodated VMs is $(3*3+4*1)*2+3*2=32$ under this bandwidth allocation strategy. If one reserves 3 units of bandwidth for the server b and server d, and 1 units of bandwidth for the server a and server c. Server f gets the remaining 2 units of bandwidth. Then the maximum total size of accommodated VMs can be $(3*1+4*3)*2+4*2=38$, which is larger than the former allocation strategy. The above example shows that the bandwidth allocation for each server has significant influence to the total resource capacity. In the following, we discuss the optimal bandwidth allocation in the network.

Lemma 1. *An efficient solution will allocate the server that has a larger CPU capacity with a larger bandwidth capacity.*

Proof: Assume that B_{total} is the total uplink bandwidth of all the core switches, and m is the total number of servers. Each server is associated with a CPU capacity C_i. We may assume that the CPU capacities satisfy $C_1 \geq C_2 \cdots \geq C_m$. Let B_i denote the bandwidth allocated to the server s_i, we have $\sum_{i \in [1,m]} B_i = B_{total}$. The maximum total resource capacity of the datacenter is $\sum_{i \in [1,m]} B_i C_i \leq B_1 C_1 + C_2(\sum_{i \in [2,m]} B_i) = B_1 C_1 + C_2(B_{total} - B_1) = C_2 B_{total} + (C_1 - C_2)B_1$. Considering $C_1 \geq C_2 \cdots \geq C_m$, we can draw the conclusion that, to maximize total resource capacity, we should keep $B_1 \geq B_2 \cdots \geq B_m$. The proof is completed. ■

According to Lemma 1, we reserve bandwidth for each server in the network according to the following steps.

Step 1: Construct an extended network $\mathcal{G}'$ according to the following procedures. Create a virtual source node v_s and a

virtual sink node v_t. Add the virtual source node and sink node to the original network $\mathcal{G}$ by connecting each core switch to the sink node, and connect each server to the source node. Set the bandwidth capacity of the link between sink node v_t and core switch v_i as the uplink bandwidth of v_i to the Internet, set the bandwidth capacity of the link between source node v_s and each server s_i to infinity. Set the unit flow cost of the link connecting source node v_s and each server s_i as C_i. Set the unit flow costs of other links in the network $\mathcal{G}'$ to 0.

Step 2: Compute the maximum cost maximum flow with the following procedures. First of all, use the additive inverse of the cost of each link as the new cost of the link. Then, use the Successive shortest path and capacity scaling method to compute the minimum cost maximum flow [21]. Note that the termination condition when the successive shortest path and capacity scaling method is applied should be changed. The algorithm terminates when the length of the shortest path is no longer negative.

Step 3: Allocate bandwidth to servers according to **Step 2**.

Lemma 2. *The above algorithm gives an upper bound of the optimal solution of the BG-VMP problem.*

Proof: According to the well-known minimum cost flow theory [21] we know that the cost of the flow computed by the above algorithm is the maximum. Since only the server nodes of the network have costs and the cost of each server is the product of the server CPU capacity and the flow capacity passing through the server, allocating bandwidth for each server according to the max flow computed will give an upper bound of the optimal solution of the BG-VMP problem. ■

After reserving bandwidth for each server according to the above algorithm, the discussion on the BG-VMP problem can be classified into two cases: 1) *Homogeneous Case.* It refers to the situation that all servers have identical CPU capacities and allocated bandwidths; 2) *Heterogeneous Case.* It refers to the situations that the CPU capacity and allocated bandwidth of each server is not identical.

A. Homogeneous Case

We first consider the problem that packing VM requests into servers with the same CPU capacity and bandwidth capacity (denoted by P1) and give the following definition and lemma.

Definition 4. *[22] An input sequence is termed accomodating if there exists an optimal offline algorithm that can accept all items.*

Lemma 3. *For m identical servers, and an accommodating input, there exists an offline algorithm for the P1 problem that achieves $1/2$ approximation in linear time.*

Proof: Since each server has the same CPU capacity and is reserved for the same bandwidth capacity, we assume that both the CPU capacity and the bandwidth of each server are 1. Let c_i' and b_i' represent the normalized CPU demand and the normalized bandwidth demand of each VM r_i, respectively. Given an accommodating VM sequence, we first use the 2-approximation algorithm proposed by Kellerer *et. al.* [23] to pack all the VMs into at most $2m$ servers. The time complexity of this algorithm is $O(n \log n)$. By choosing the m servers with the most size of VMs, we can place no less than $m/2m$ VMs in linear time. Thus, we get a $1/2$ approximation. ■

Combining Lemma 2 and 3, we obtain the following.

Theorem 2. *For an accommodating input, there exists an offline algorithm for this homogenous BG-VMP problem that achieves $1/4$ approximation in linear time.*

Proof: Let Q and Q' represent the optimal solutions for BG-VMP and P1, respectively. Let I denote the solution of our algorithm. Let H denote the maximum total resource capacity. It is easy to verify that the inequality $\frac{H}{2} \leq Q' \leq Q \leq H$ is satisfied. Thus, we have $\frac{I}{Q} \geq \frac{I}{H} \geq \frac{I}{2Q'} \geq \frac{1}{2*2} = \frac{1}{4}$. ■

B. Heterogeneous Case

The inconsistency of resources on servers brings in an additional complexity. For m servers, each server has a CPU capacity (C_i) and a bandwidth capacity (B_i). Let C_m and B_m denote the minimum value among set $\{C_i | 1 \leq i \leq m\}$ and set $\{B_i | 1 \leq i \leq m\}$, respectively. For brevity, we define some notations that will be used in the following part. Let $\alpha_i = \min\{\lfloor \frac{C_i}{C_m} \rfloor, \lfloor \frac{B_i}{B_m} \rfloor\}$ and $\beta_i = \max\{\lceil \frac{C_i}{C_m} \rceil, \lceil \frac{B_i}{B_m} \rceil\}$. Let $\overline{\alpha}$ and $\overline{\beta}$ denote the average value of server CPU capacity and allocated bandwidth, respectively (i.e., $\overline{\alpha} = \frac{\sum_{i\in[1,m]} \alpha_i}{m}$ and $\overline{\beta} = \frac{\sum_{i\in[1,m]} \beta_i}{m}$). Let γ represent the ratio of $\overline{\beta}$ and $\overline{\alpha}$ (i.e., $\gamma = \frac{\overline{\beta}}{\overline{\alpha}}$). Here, γ is no less than 1. We first consider the problem that packing VM requests into servers with heterogeneous CPU capacity and bandwidth (denoted by P2).

Lemma 4. *For an accommodating input, there exists an offline algorithm for the P2 problem that achieves $1/2\gamma$ ($\gamma \geq 1$) approximation in linear time.*

Proof: As defined above, the minimum CPU capacity among all the servers is C_m, and the minimum allocated bandwidth among all the servers is B_m. For each VM r_i, let c_i' denote the normalized CPU capacity of C_m, and let b_i' denote the normalized bandwidth requirement of B_m. Both c_i' and b_i' are positive and no larger than 1.

Given an accommodating VM sequence of the original server set $\mathcal{S}$, we assume that the minimum number of unit servers with the resource vector $< 1, 1 >$ needed to pack all the VMs is OPT_{unit}. Firstly, we use the 2-approximation algorithm proposed by Kellerer *et. al.* [23] to pack all the VMs into at most $2OPT_{unit}$ unit servers. Then, we pick up $\sum_{i\in[1,m]} \alpha_i$ unit servers with the most sizes of VMs. Finally, for each server s_i we allocate all the VMs that have been packed in α_i unit servers to it. Since $\alpha_i = \min\{\lfloor \frac{C_i}{C_m} \rfloor, \lfloor \frac{B_i}{B_m} \rfloor\}$, all the VMs in α_i unit servers can be covered by the server s_i. Thus the above is a feasible placement.

Now we prove that the above placement yields $1/2\gamma$ approximation in linear time. The approximation of the above algorithm is no less than $\frac{\sum_{i\in[1,m]} \alpha_i}{2OPT_{unit}}$. Since $OPT_{unit} \leq \sum_{i\in[1,m]} \beta_i$, thus, $\frac{\sum_{i\in[1,m]} \alpha_i}{2OPT_{unit}} \geq \frac{\sum_{i\in[1,m]} \alpha_i}{2\sum_{i\in[1,m]} \beta_i} = \frac{\overline{\alpha}}{2\overline{\beta}} = 1/2\gamma$. The proof has been completed. ■

Algorithm 1 SortFirstFit

Input: A data center network $\mathcal{G}$ and a new arriving VM r.
Output: The placement of VM r.

1: Arrange the servers in descending order according to the CPU capacity.
2: **for** $s_i \in \mathcal{S}$ **do** ▷ For each server s_i.
3: Compute the maximum possible bandwidth B_i.
4: **if** $C_i' \geq c_i$ and $B_i \geq b_i$ **then**
5: Place the VM r in the server s_i.
6: Break.
7: Update the residual network.

Combine Lemma 2 and Lemma 4, we can get Theorem 3 as follows. The proof is similar with Theorem 2, We omit the proof of Theorem 3 due to the space limit.

Theorem 3. *For an accommodating input, there exists an offline algorithm for this heterogeneous BG-VMP problem that achieves* $1/4\gamma$ *(*$\gamma \geq 1$*) approximation in linear time.*

IV. Online Heuristics

Motivated by the theoretical analysis for the offline BG-VMP problem, we present two main principles for the online version of the BG-VMP problem.

Principle 1. The utilizations of both CPU capacity and available bandwidth should be jointly considered.

Principle 2. The optimal VM placement strategy should allocate as much bandwidth as possible to the servers with larger CPU resources.

Based on the two principles, the online BG-VMP problem is accomplished by two simple and practical heuristics. The first one is SortFirstFit. (in Algorithm 1) This algorithm always places the arriving VM into the server with maximum CPU capacity which has enough residual CPU capacity and bandwidth. The second one is SortWorstFit. (in Algorithm 2) This algorithm always places the arriving VM into the server with maximum residual size. (as defined in Algorithm 2, line 4) The maximum possible bandwidth in both algorithms are computed by solving a maximum flow problem.

Algorithm 2 SortWorstFit

Input: A data center network $\mathcal{G}$ and a new arriving VM r.
Output: The placement of VM r.

1: **for** $s_i \in \mathcal{S}$ **do** ▷ For each server s_i.
2: Compute the maximum possible bandwidth B_i.
3: **if** $C_i' \geq c_i$ and $B_i \geq b_i$ **then**
4: Mark the residual size of s_i with $C_i' * B_i$.
5: Place the VM r in the server with maximum residual size.
6: Update the residual network.

V. Evaluation

Settings. The three-tier hierarchical network adopted here is a 10-ary 3-tree [24] composed of 1000 servers. The bandwidth of each link between a level 2 switch and a server follows a uniform distribution with the range of $[2, 10]$. We define that the oversubscription ratio of a switch is the ratio of the aggregate uplink bandwidth to the aggregate downlink bandwidth. The oversubscription ratio of each level 0, 1 and 2 switch is set to 1, 1 and 0.7, respectively. The CPU capacity of each server follows a uniform distribution with the range of $[2, 10]$. The CPU demand and bandwidth demand information of input VMs used for the evaluations are generated data sets and real trace-driven data sets. Three generated data sets are used in this paper to evaluate the different aspects of the proposed heuristics. A real trace-driven data set is also used in this evaluation. Specifically, the CPU resource information is from Google cluster management data [25] and the bandwidth resource information is from a data warehouse hosted by IBM Global Services [7]. We evaluate the two heuristics proposed in this paper, i.e., SortFirstFit and SortWorstFit. We also evaluate the performance of Random, with FirstFit and WorstFit as the benchmarks. Note that the latter two algorithms only consider CPU capacity. Since solving the BG-VMP problem in a large input instance is NP-hard, the optimal solution is not available to obtain as a baseline. We vary the number of servers to see the total accommodating VMs of each heuristic in this paper to show the effectiveness of the proposed online algorithms.

Measuring the algorithm performance and the impact of arriving VM sequence. We evaluate the performance of the proposed heuristics under two types of arriving VM sequences firstly. The first type of arriving VM sequence consists of a sequence of small VMs followed by a sequence of large VMs; The second type of arriving VM sequence consists of a sequence of large VMs followed by a sequence of small VMs. The results of the first data set are shown in Figure 2(a). The input sequence consists of 1k VMs of size $\langle 0.2, 0.4 \rangle$ followed by a sequence of VMs of size $\langle 0.8, 1.0 \rangle$. From the results, we can see that SortFirstFit and SortWorstFit work much better than the other heuristics. More specifically, the FirstFit works slightly better than the WorstFit in this situation. When they are provided with the same amount of servers, the former can accommodate more VMs. The Random heuristic has the worst performance. The results of the second data set are shown in Figure 2(b). The input sequence consists of 2k VMs of size $\langle 0.8, 1.0 \rangle$ followed by a sequence of VMs of size $\langle 0.2, 0.4 \rangle$. It can be seen from the results that the Random heuristic is still the worst. However, in this situation, the performance of all the heuristics except the Random heuristic are close. Combining Figure 2(a) and Figure 2(b), we can draw the conclusion that processing sorting before the placement may make the placement more efficient.

Measuring the impact of hardware heterogeneity. The results of the third data set are shown in Figure 2(c). In this situation, we keep the amount of servers fixed and vary the CPU capacity and the bandwidth of each link between a level 2 switch and a server to see the effect on the performance of each heuristic. The network involves 1k servers. The CPU capacity and the bandwidth of each link between a level 2 switch and a server follow a uniform distribution with the

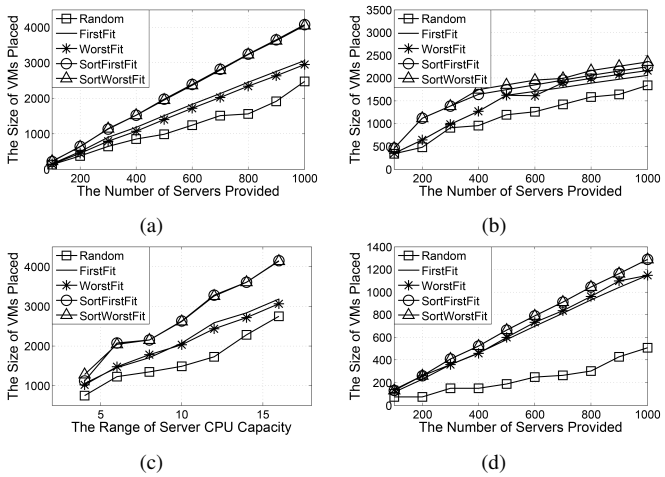

Fig. 2. The number of VMs that can be accommodated when providing different number of servers under different placement strategies with a) Small VMs followed by large VMs and b) Large VMs followed by small VMs. c) The number of VMs that can be accommodated given different hardware heterogeneity settings under different placement strategies. d) The number of VMs that can be accommodated providing various number of servers under different placement strategies using the real traces.

range of $[2, x]$, where x varies in range $[4, 16]$. Both the CPU resource and bandwidth resource requirements of input VMs follow the uniform distribution with the range of $[0.1, 1]$. It can be seen that when the average server CPU capacity and the average link bandwidth increase, the number of accommodated VMs increases. Besides, the SortFirstFit and the SortWorstFit work better than the FirstFit and the WorstFit. What's more, the advantages of sorting will be more significant when the difference of resources of servers in the network are larger.

Testing with real traces. The results of the real trace-driven data set are shown in Figure 2(d). The CPU resource requirements of the VMs in the input sequence varied between 0.25, 0.5, and 1 with the probabilities of 0.99%, 92.67%, and 6.34%, respectively [25]. 80% of the input VMs have a bandwidth requirement less than 0.1 and 4% of them have a bandwidth requirement larger than 1 [7]. It can be seen from the results that the Random heuristic has the worst performance. The SortFirstFit and the SortWorstFit perform better than the FirstFit and the WorstFit. The results in this situation verify that the performances of the five heuristics using trace-driven data sets are in line with those using the above three generated data sets.

VI. Conclusion

In this paper, we studied the the problem of placing VMs in cloud data centers to maximize simultaneously the resource utilizations of both the server and the network. The problem is important as the overall efficiency in a data center is dictated by multiple resources and one may contradict another if optimizations are carried out independently in each of the resource dimension. Depending on whether the hardware resources in a data center are homogeneous or not, we develop bounded approximation algorithms in the offline scenario, while providing efficient heuristics based on the insights from the offline algorithm design in the general online scenarios, for each of the two cases. The effectiveness of the algorithms is verified through theoretical analysis and extensive simulations.

References

[1] M. Mishra and A. Sahoo, "On theory of VM placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *IEEE CLOUD*, 2011.

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

[3] X. Jin, F. Zhang, L. Wang, S. Hu, B. Zhou, and Z. Liu, "Joint optimization of operational cost and performance interference in cloud data centers," *TCC*, vol. PP(99), 2015.

[4] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *J. Network Syst. Manage.*, vol. 23, no. 3, pp. 567–619, 2015.

[5] Z. Á. Mann, "Allocation of virtual machines in cloud data centers - A survey of problem models and optimization algorithms," *ACM Comput. Surv.*, vol. 48, no. 1, p. 11, 2015.

[6] J. Arjona, A. Fernández, M. A. Mosteiro, C. Thraves, and L. Wang, "Power-efficient assignment of virtual machines to physical machines," *Future Generation Comp. Syst.*, vol. 54, pp. 82–94, 2016.

[7] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM*, 2010.

[8] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, "Multi-resource packing for cluster schedulers," in *SIGCOMM*, 2014.

[9] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[10] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *SIGMETRICS PER*, vol. 41, no. 3, pp. 107–112, 2013.

[11] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu, "Greendcn: A general framework for achieving energy efficiency in data center networks," *IEEE JSAC*, vol. 32, no. 1, pp. 4–15, 2014.

[12] D. Li and J. Wu, "Reducing power consumption in data centers by jointly considering VM placement and flow scheduling," *Journal of Interconnection Networks*, vol. 15, no. 1-2, 2015.

[13] N. Patel and G. Patel, "Vm placement of multidimensional resources using cartesian coordinates based approach," in *NUiCONE*. IEEE, 2015, pp. 1–5.

[14] L. Wang, A. F. Anta, F. Zhang, J. Wu, and Z. Liu, "Multi-resource energy-efficient routing in cloud data centers with network-as-a-service," in *ISCC*, 2015, pp. 694–699.

[15] Q. Liu, G. Wang, J. Wu, and W. Chang, "User-controlled security mechanism in data-centric clouds," in *HPCC*, 2015, pp. 647–653.

[16] S. Hu, W. Bai, K. Chen, C. Tian, Y. Zhang, and H. Wu, "Providing bandwidth guarantees, work conservation and low latency simultaneously in the cloud," in *INFOCOM*, 2016.

[17] "Cisco data center infrastructure 2.5 design guide," 2007.

[18] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008.

[19] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *NSDI, April 11-13, Cambridge, Massachusetts, USA*, 2007.

[20] L. Epstein and L. M. Favrholdt, "On-line maximizing the number of items packed in variable-sized bins," *Acta Cybern.*, vol. 16, no. 1, pp. 57–66, 2003.

[21] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, no. 2, pp. 248–264, 1972.

[22] J. Boyar, K. S. Larsen, and M. N. Nielsen, "The accommodating function: A generalization of the competitive ratio," *SIAM J. Comput.*, vol. 31, no. 1, pp. 233–258, 2001.

[23] H. Kellerer and V. Kotov, "An approximation algorithm with absolute worst-case performance ratio 2 for two-dimensional vector packing," *Oper. Res. Lett.*, vol. 31, pp. 35–41, 2003.

[24] F. Petrini and M. Vanneschi, "k -ary n -trees: High performance networks for massively parallel architectures," in *IPDPS, April 1-5, Geneva, Switzerland*, 1997.

[25] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Technical Report, 2011.