DVMP: Incremental Traffic-aware VM Placement on Heterogeneous Servers in Data Centers

Dan Li, Syed Shah-e- Mardan Ali Rizvi, Wu He, Fangxin Wang

Tsinghua University

***Abstract*—As the tremendous momentum cloud computing has grown, the modern data center networks are facing challenge to handle the increasing traffic demand among virtual machines (VMs). Simply adding more switches and links may increase network capacity but at the same time increase the complexity and infrastructure cost. Thus, intelligent VM placement has been proposed to reduce the intra-DC traffic. Prior solutions model the traffic-aware VM placement problem as a Balanced Minimum K-cut Problem (BMKP). However, the assumptions of "once-for-all" VM placement on physical servers with equal VM slots are often not realistic in practical data centers, and thus the naive BMKP model may lead to suboptimal placement solutions.**

In this work, we revisit the problem by considering the server heterogeneity and propose an incremental traffic-aware VM placement algorithm. Given that the BMKP model cannot be directly applied, we make a number of transformations to re-establish the model. First, by introducing pseudo VM slots on physical servers with less VM slots, we allow the number of available VM slots of each server to be different. Second, pseudo edges with infinite costs are added between existing VMs, and thus previously deployed VMs on the same physical server will still be packed together. Third, a change on the number of pseudo VM slots is applied, so that existing VMs placed on different physical servers will still be separated. In this way, we reduce the problem to a new BMKP problem, which results in a much better solution. The evaluation results show that DVMP can reduce up to 28%, 39% and 55% traffic compared with naive BMKP model, greedy VM placement and random VM placement, respectively.

I. Introduction

With the proliferation of cloud computing, more and more applications are deployed in the data centers. Moreover, the cloud applications are usually bandwidth hungry, for example, MapReduce [1], HDFS [2], which shuffle a large amount of data for computation. One of the challenges that the data center operators face is the increasing traffic demand among VMs, *i.e.,* the 'east-western' traffic within the data center. In the past few years, numerous efforts have been spent to address this challenge in both industry and academia. Advanced network topologies with more switches and links are designed to expand the network capacity, such as BCube [3], Fat-Tree [4], VL2 [5] and FiConn [6], with an extra infrastructure cost. In-network computation [7] and in-network caching [8], [9] solutions are also proposed to reduce the traffic pressure to the network, which require modifications to the switch functionalities and thus is hard to deploy.

Complimentary to these fundamental changes discussed above, the traffic-aware virtual machine (VM) placement was suggested to optimize the bandwidth utilization given the available resources without increasing the CAPEX [10], [11]. In today's data center, the basic computation unit for resource allocation is a VM with virtualized CPU and memory resources. Data center operators have high flexibility in choosing the physical server to place the VMs in order to achieve a diverse set of goals, such as consolidating multiple VMs in a physical server to improve resource multiplexing, migrating VMs to different physical machines for load balance, *etc.*. Traffic-aware VM placement is one way to assign VMs to minimize the total bandwidth consumption. More specifically, some VMs have more traffic between them, so placing them into the same physical server or physically closer servers can help reduce the total amount of traffic carried by the network.

Prior work models the traffic-aware VM placement problem as a balanced minimum k-cut problem (BMKP) [10], [11]. In this model, the VMs of a job are divided into k groups, each of which is mapped to a physical server with equal number of VMs. However, the hidden assumption of the naive BMKP model may be broken in practical data centers, which brings considerable challenges to solve the problem. First, the naive BMKP model assumes a "once-for-all" VM placement problem. In other words, it assumes that the requests for all the VMs and their traffic matrixes in the entire data center are given at once as input, and then we use the BMKP algorithm to find the optimal placement. In reality, the computation jobs dynamically arrive one after another, and the "once-for-all" placement approach may lead to suboptimal placement result. On one hand, it does not consider VM slots taken by the existing jobs. On the other hand, the new job may go beyond communicating among the VMs of itself, and may communicate with the previously deployed VMs. For instance, the new VMs need to read data from and write data to the existing distributed file system. As we will show in Section II-B, the traffic volume between new VMs and existing VMs can be even larger than that within new VMs, and thus cannot be ignored.

Second, the naive BMKP model supposes that all the physical servers have the same number of VM slots. But in practice, the number of VM slots in the physical servers can be quite different due to two reasons. On one hand, VM slots in some physical servers can be partially occupied by existing jobs. On the other hand, due to the purchase from different vendors and rapid hardware innovations, data center servers are usually equipped with different hardware configurations [12], [13] and different capacities to host VMs. Both the factors result in different numbers of available VM slots in physical servers when placing the VMs for a job, which challenges the naive BMKP model too.

In this paper, we revisit the traffic-aware VM placement problem by breaking the assumptions on prior models and supporting incremental traffic-aware VM placement on heterogeneous servers in data centers, and the new algorithm is called *DVMP*. Given that servers are heterogeneous and the naive BMKP model cannot be directly applied, DMVP makes a number of transformations to re-establish the model. First, pseudo VM slots are added on physical servers with less VM slots, so as to make the physical servers equivalent. Second, pseudo edges with infinite costs are added between existing VMs, and thus previously deployed VMs on the same physical server will still be packed together. Third, a change on the number of pseudo VM slots is applied, so that existing VMs placed on different physical servers are still separated. In this way, we formulate this problem to a new variant of BMKP problem which improves the results of traditional method.

We conduct extensive simulations to study the performance of DVMP, which is compared to random placement, greedy placement and state-of-the-art naive BMKP placement. Results suggest that DVMP can save up to 55% traffic relative to random placement, 39% relative to greedy placement and 28% relative to state-of-the-art naive BMKP placement. Besides, simulations with other parameters also prove the effectiveness of DVMP in practical scenarios.

The rest of this paper is organized as follow. Section II states the problem. Section III describes the solution for the problem. Section IV evaluates the effectiveness of the solution by simulation. Related work is discussed in section V and section VI concludes the paper.

II. Background and Problems

In this section, we introduce the problem with existing model and discuss the two unrealistic assumptions we identify in this work.

A. Existing Models

In the traffic-aware VM placement problem, the input is the set of VMs to place and the amount of traffic between VM pairs, while the output is the mapping from VMs to the physical servers. The problem is modeled as a graph partition problem in prior works [10], [11]. Specifically, the VMs and their communications form a graph G, where the nodes represent VMs of a job, and the edges represent communication between VMs. The weight of each edge indicates the traffic demand between two VMs. Placing the VMs onto m physical servers is equivalent to partitioning G into m subgraphs. The optimization goal is to minimize the total weight of edges across subgraphs. Several available graph partition models in the literature [14], [15] can be used to formulate this problem.

(1) *Minimum K-cut Problem (MKP):* Finding a set of edges, after removing them, G is partitioned into k subgraphs: $g_1, g_2, ..., g_k$, and the sum of the weights of these removed edges is minimized. Note that there is no constraint on the size of each subgraph, *i.e.,* $1 \leq |g_i| \leq |V| - k + 1, 1 \leq i \leq k$.

(2) *Balanced Minimum K-cut Problem (BMKP):* a constraint is added to MKP, *i.e.,* sizes of subgraphs are strictly equal: $|g_i| = \frac{|V|}{k}, 1 \leq i \leq k$.

(3) *(k,ν)-BMKP:* This model is similar to BMKP, but it relaxes the constraint for subgraph sizes. For the k partitioned subgraphs $g_1, g_2, ..., g_k$, only an upper-bound constraint is introduced to make them roughly equal: $|g_i| \leq \nu\frac{|V|}{k}(1 \leq i \leq k, \nu \geq 1)$. Note that BMKP is a special case of (k,ν)-BMKP where $\nu = 1$.

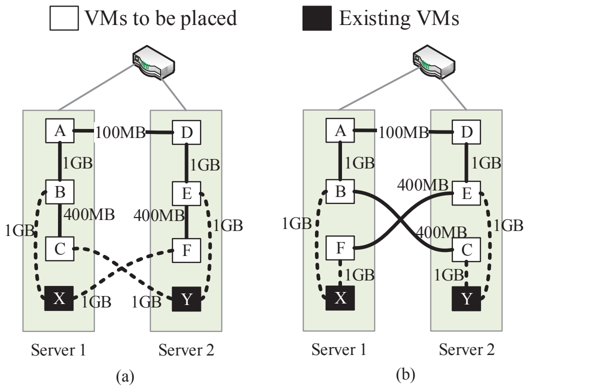

Fig. 1. Example to demonstrate the problem with pure BMKP-based VM placement. New VM A∼F are waiting to be placed on two servers. Denote $\xi(\cdot)$ as the traffic demand between new and existing VMs (dashed line), $\xi(B,X) = \xi(F,X) = \xi(C,Y) = \xi(E,Y) =$ 1GB. (a) pure BMKP-based VM placement ignoring $\xi(\cdot)$. The total network traffic is 2.1GB. (b) An optimal solution. The total network traffic is 0.9GB.

Prior works [10], [11] abstracted the traffic-aware VM placement problem as a pure BMKP problem. In what follows, we argue that there are two hidden assumptions used in the model which may not be valid in practical data centers.

B. Assumption 1: "Once-for-All" VM Placement

Existing traffic-aware VM placement schemes depend on a "once-for-all" assumption. But in a public cloud, a tenant may incrementally request for new VMs to dynamically expand his/her computing capacity. The new VMs to deploy not only communicate among themselves, but may also communicate with existing VMs. The problem is even more common in a private data center, where many dynamically assigned jobs could belong to the same user/group and hence cross-job communications exist. Our investigation of a workload from Facebook data center shows that the traffic between new VMs and existing VMs are even more than the traffic between new VMs themselves (refer to Section IV for the detail), and hence the problem is unneglectable.

As stated in Section I, rerunning the BMKP algorithm on all the VMs again whenever a new request arrives incurs unaffordable cost, while using the pure BMKP model to place the new VMs without considering their traffic with the existing VMs can result in suboptimal result. We use one example to illustrate the problem in Fig. 1. In this example, 6 VMs are to

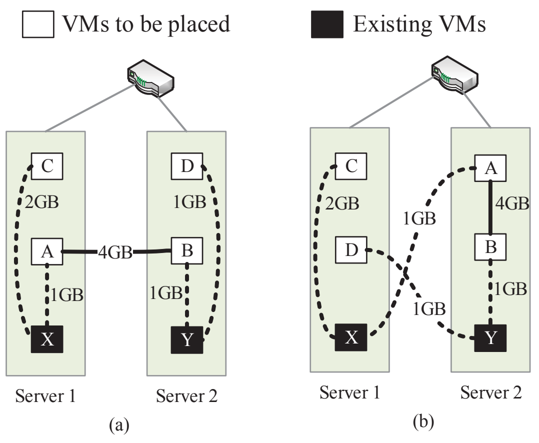

Fig. 2. Example to demonstrate the problem with greedily placing VMs close to data source. New VM A∼D are waiting to be placed on two servers. Denote $\xi(\cdot)$ as the traffic demand between new and existing VMs (dashed line), $\xi(A, X) = \xi(B, Y) = \xi(D, Y) = $ 1GB, $\xi(C, X) = $ 2GB. (a) Greedily placing VMs close to data source. The total network traffic is 4GB. (b) An optimal solution. The total network traffic is 2GB.

be placed and 2 VMs are already placed. The traffic demand between them are shown in the figure. The left solution is an outcome from running the pure BMKP algorithm directly on the new VMs. As we can see, it results in 2.3 times more network traffic compared to the right solution which is optimal.

One may argue that the new VMs can be greedily put close to the data sources in existing VMs, as many MapReduce tasks do, *i.e.,* putting a mapper to the server with the data [16]. However, the simple greedy based approach does not consider the global optimization and may lead to suboptimal solution as well. Fig. 2 demonstrates an example in which the greedy method generates 2 times more traffic than the optimal one.

As a result, in order to get a desired VM placement solution, we need to simultaneously consider the traffic between new VMs as well as the traffic between new and existing VMs.

C. Assumption 2: Equal Number of VM Slots in Servers

Prior works also assume that the servers have equal number of VM slots, and thus BMKP can be applied. However, heterogeneous configuration widely exists among data center servers due to two reasons. First, hardware innovation is rapid. It has been shown that data center owners usually incrementally purchase servers quarterly or yearly [17] to accommodate the service expansion. It is common that the servers bought at different times have different hardware profiles. The heterogeneity lies in the number of CPU cores, the memory space, *etc.* Second, for cost consideration, data center operators can buy servers from different vendors, which can also result in servers' hardware heterogeneity. The impact of server heterogeneity on data center architecture design has also been noticed in recent works [13], [12]. Given heterogeneous hardware profile of physical servers, the number of VM slots residing in servers can be different.

Moreover, as described above, computation jobs are usually incrementally deployed in data centers. When placing VMs of a new job onto the physical servers, some VM slots of the available servers may already be occupied by existing jobs. To achieve multiplexing gains, it is a common practice to mix VMs of different tenants on the same physical machine. In such cases, it leads to unequal number of available VM slots in the physical servers when placing VMs of a new job.

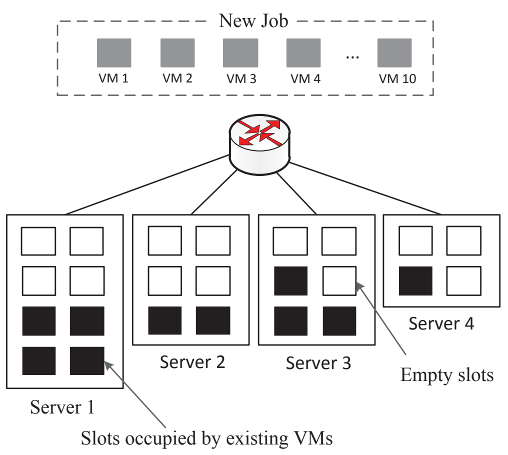

Fig. 3. An example scenario of placing $n = 10$ new VMs to 4 heterogeneous servers. The total number of VM slots on each server is 8, 6, 6, 4, respectively; and the number of existing VMs on each server is 4, 2, 3, 1, respectively.

Both factors result in server imbalance in VM placement, *i.e.,* the varied number of VM slots, which violates the assumptions of the pure BMKP model too.

III. DVMP Design

In this section, we design the *DVMP* algorithm to incrementally place VMs on heterogeneous servers, with the optimization goal of minimizing overall network traffic. The major notations used in this section is listed in Table I.

TABLE I
NOTATIONS

n	Number of new VMs to place
m	Number of physical servers
Y	Number of previously deployed VMs that new VMs communicate with
v_i	The i-th new VM to place, $v_i \in V = \{v_1, v_2, ..., v_n\}$
p_j	The j-th physical server, $p_j \in P = \{p_1, p_2, ..., p_m\}$
e_i	The i-th previously deployed VM, $e_i \in E = \{e_1, e_2, ..., e_Y\}$
C_i	Number of VMs slots provided by server p_i
T_i	Number of occupied slots on server p_i
S_i	Number of available slots on server p_i
B_i	Number of marked nodes corresponding to server p_i
L	Number of dummy nodes
Z	Size of each subgraph (group) in the translated BMKP problem
F_{ij}	Size of traffic interaction between VM v_i and e_j.
M_{ij}	Size of traffic from VM v_i to VM v_j

A. Model Selection

We first take Fig. 3 as an example, where 10 new VMs are waiting to be deployed on 4 physical servers and the new VMs have communications with 10 previously deployed VMs in the servers. In order to encompass all the new VMs and existing VMs (with traffic interaction to new VMs), we get a graph G with 20 nodes. A desired VM placement scheme corresponds to a partition of G into 4 subgraphs: g_1, g_2, g_3 and g_4, with $|g_1| \in [4, 8]$, $|g_2| \in [2, 6]$, $|g_3| \in [3, 6]$ and $|g_4| \in [1, 4]$. Besides, each existing VM $e_i (1 \leq i \leq Y)$ should still remain in its original server $\pi(e_i)$. We discuss the possible models in the literature that may be applied to this problem.

1) MKP: The minimum K-cut model is not suitable for our problem, since it has no constraint on the size of subgraphs at all. As a result, G may be partitioned into subgraphs with undesired sizes, which does not fit the number of VM slots of each physical server.

2) BMKP: Assume the naive BMKP model is applied to our problem, *i.e.,* partitioning G into 4 subgraphs with each containing 5 nodes. It does not work either, since server 4 only has 4 VM slots.

3) (k, ν)-BMKP: We then check the (k, ν)-BMKP model, in which a single upper-bound constraint on all the subgraph sizes is enforced. The first question is how to appropriately set the upper bound, *i.e.,* $|g_i| \leq \nu \frac{|V|}{k}, 1 \leq i \leq k, \nu \geq 1$. If we set the upper bound as $\nu \frac{|V|}{k} = 5$, it will be the same as BMKP, which is inappropriate. If setting $\nu \frac{|V|}{k} \geq 6$, it also fails for server 4.

In a word, all the three models above cannot be directly applied to solve our problem. To seek for an optimal solution, we have to make transformations of the problem to make an existing model applicable. Noting that MKP and (k, ν)-BMKP suffer from indeterminate size of subgraphs, which may be invalid when mapping them to physical servers, we focus on converting our problem to BMKP. The subgraph sizes of BMKP is $(\frac{|V|}{k})$, as described in section III-C.

B. Design Challenges

We need to address the following challenges in model transformation.

Making the Number of Nodes in the Graph Equal: This challenge is straightforward. The numbers of available VM slots in the physical servers are different but we need to make equal number of nodes in the graph of the new BMKP model.

Packing Existing VMs on the Same Server in the Same Subgraph: A general graph partition algorithm treats all nodes equally, and only focuses on finding a k-cut with minimum weight. So existing VMs on the same physical server may be partitioned into different subgraphs, making it infeasible to map the resulting subgraphs to physical servers. However, in the new BMKP model, all existing VMs $\{e_i | \pi(e_i) = p_t, 1 \leq i \leq Y\}$ on the same server $p_t (1 \leq t \leq m)$ should be packed together in the same subgraph.

Separating Existing VMs on Different Servers into Different Subgraphs: Due to the same reason above, a general graph partition algorithm may also lead existing VMs on different servers being merged into the same subgraph, which violates the requirement, too. The new model should tackle this problem and separate existing VMs on different servers into different subgraphs.

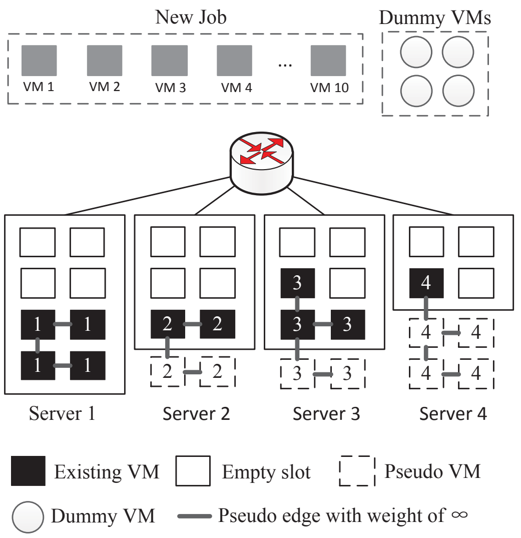

Fig. 4. An example of how to deal with server heterogeneity: adding *pseudo slots* occupied by *pseudo VMs* for each physical server without affecting placement of new VMs. The pseudo VMs are regarded the same as existing VMs except that they don't have any traffic.

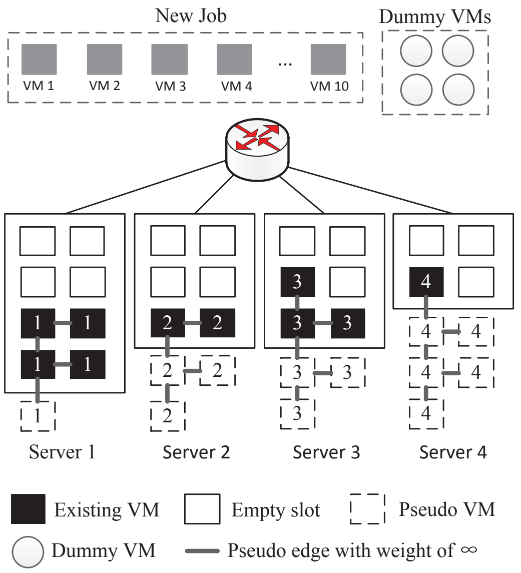

Fig. 5. By applying a change on the number of *pseudo slots (VMs)*, we can always separate existing VMs on different servers into different subgraphs.

C. Problem Transformation

Besides the new VMs to deploy and existing VMs, we introduce two concepts to help transform the problem.

Pseudo Slot and Pseudo VM: For each server $(p_i, 1 \leq i \leq m)$ with S_i available VM slots and T_i occupied slots, we can just regard it as a server with S_i available slots and $B_i \geq T_i$ occupied slots. Note that this transformation does not affect the placement of new VMs, since there are still S_i available slots on p_i. The extra $B_i - T_i$ slots are imagined and in fact do not exist, so we call them *pseudo slots*. Correspondingly, the imagined VMs occupying these pseudo slots are called *pseudo VMs*, which can be regarded as existing VMs except that they do not send or receive any traffic. In this way, we can change the number of VM slots on physical servers without affecting the optimal placement solution for the new VMs.

Pseudo Edge: To help transform the problem we also add imaginary additional edges between VMs and these edges are called *pseudo edges*. Note that the pseudo edges added should not affect the optimal VM placement solution.

Dummy VM: When the overall available slots ($\sum_{i=1}^{m} S_i$) are more than the new VMs, *i.e.,* $n \leq \sum_{i=1}^{m} S_i$, we need to choose n from the available slots for the new VMs, and this can be difficult in practice due to the abundance of optional choices. As an alternative way, we introduce *dummy VMs* to avoid this problem. A dummy VM is regarded as a new VM to deploy except that it does not send or receive any traffic. By introducing $L = \sum_{i=1}^{m} S_i - n$ dummy VMs, we place $\sum_{i=1}^{m} S_i$ VMs on $\sum_{i=1}^{m} S_i$ slots, and the optimal placement solution is equivalent to that without them, as a result of their "dummy" feature. For the example in Fig. 3, we add 4 dummy VMs and regard them as new VMs (Fig. 4).

By introducing pseudo slots, pseudo VMs and dummy VMs, DVMP addresses the three challenges above in the following ways. First, we add pseudo slots to each server to make them hold equal number of VM slots. Fig. 4 shows an example of this. By adding 2, 2, 4 pseudo slots for server 2, 3 and 4 respectively, servers are made equivalent to enable BMKP's applicability. For pseudo VMs added to server p_i, we mark them with i (see Fig. 4) to show that they should be mapped to p_i however the new VMs are placed. Similarly, the existing VMs that are previously deployed on server p_i should be mapped to it as well. In the translated problem, pseudo VMs are regarded as existing VMs, thus hereafter we make no distinction between them, and uniformly regard them as existing VMs marked with the corresponding server identity. Consequently, we can gracefully deal with the first challenge.

To address the second challenge and prevent graph partition algorithms from partitioning existing VMs on the same server away, we connect the existing VMs on each server with pseudo edges with the weight of ∞ (the topology of them is a tree for simplicity, as shown in Fig. 4), hence the graph partition algorithms would never cut them away. Otherwise, the corresponding k-cut will have a weight of ∞, which will not be the desired solution.

The problem of merging existing VMs on different servers may occur only when the total number of nodes with mark i and j ($1 \leq i, j \leq m$) is within the subgraph size. Take Fig. 4 for example, the numbers of nodes with mark $i = 1$ and $j = 2$ are both 4, and the subgraph size is 8, hence it's possible for the 8 marked nodes to be merged into the same subgraph. To solve this problem, we apply a change (*i.e.,* δ) on the number of pseudo slots for each server in the translated problem. Shown by Fig. 5, if we increase the number of pseudo slots only by one ($\delta = 1$), the problem is solved. The subgraph size is 9 while the total number of nodes with mark 1 and 2 is 10, so the 1-marked nodes and 2-marked nodes cannot be merged into the same subgraph. The following theorem guarantees that we can always find a feasible δ to handle this

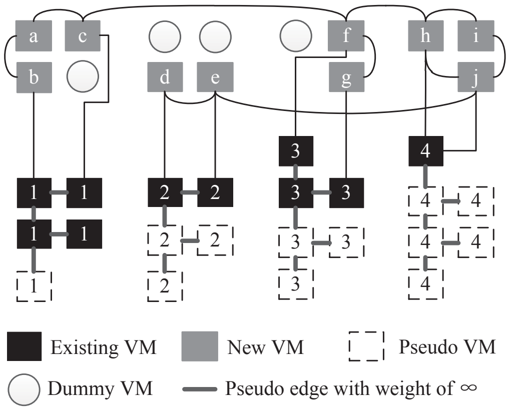

Fig. 6. An example of Alg. 2's output.

challenge.

Theorem 1. *In the translated problem, if* $\delta = 2S_{max} - C_{max} + 1 \leq 0$*, we can always separate existing VMs on different servers into different subgraphs, where* C_{max} *is the maximum number of VM slots provided by the servers. Otherwise, by adding* δ *extra pseudo slots for each server, the same goal can be achieved.*

Proof: In the translated problem, pseudo slots are added for each server to make them hold equal number of slots, *i.e.,* C_{max}. We can prove the theorem by contradiction.

1) $\delta \leq 0$*:* In this case, the size of each subgraph is C_{max}, including S_i available slots and B_i occupied slots (existing VMs). If existing VMs on server p_i and p_j ($1 \leq i, j \leq m$) are in the same subgraph, the size of the resulting subgraph is at least $B_i + B_j$. We can have $C_{max} \geq 2S_{max} + 1$ from $\delta \leq 0$. Hence, there is $B_i + B_j = 2C_{max} - S_i - S_j = C_{max} + (C_{max} - S_i - S_j) \geq C_{max} + (S_{max} - S_i) + (S_{max} - S_j) + 1 > C_{max}$. In other words, the size of the resulting subgraph is even larger than C_{max}, which leads to a contradiction. Hence, it's impossible for existing VMs on different servers to be merged into the same subgraph.

2) $\delta > 0$*:* By adding δ extra pseudo slots for every server, the number of slots on them is still identical. Denote the consequent subgraph size as Z, we have $Z = C_{max} + \delta = 2S_{max} + 1$. Similarly, we have $B_i + B_j = 2Z - S_i - S_j = Z + (Z - S_i - S_j) = Z + (S_{max} - S_i) + (S_{max} - S_j) + 1 > Z$. As a result, the existing VMs on different servers can never be merged into the same subgraph. □

By successfully transforming the problem of incremental VM placement on heterogeneous servers into a new BMKP model, we can use typical balanced minimum k-cut algorithms such as [14] to find a desired solution.

D. Algorithm

We then describe the DVMP algorithm. Alg. 1 shows the overall process. First, a graph G is constructed by Alg. 2 (line 1), which enforces the mechanisms discussed previously. Second, the BalancedMinKCut(.) algorithm is invoked to partition G into m equal sized subgraphs $\{g_1, g_2, ..., g_m\}$

Algorithm 1 DVMP Algorithm

```
Input: $V$={$v_1$,$v_2$,...,$v_n$}:new VMs
    $E$={$e_1$,$e_2$,...,$e_Y$}:existing VMs
    $P$={$p_1$,$p_2$,...,$p_m$}:physical servers
    $S_i$:Number of available slots on $p_i$
    $T_i$:Number of relevant existing VMs on $p_i$
    $M$:$M_{ij}$ is the traffic from $v_i$ to $v_j$
    $F$:$F_{ij}$ is the traffic between $v_i$ and $e_j$
Output: $\phi$(VM placement scheme)
1: $G \leftarrow$ CreateGraph($V, E, P, S_i, T_i, M, F$);
2: $\{g_1, g_2, ..., g_m\} \leftarrow$ BalancedMinKCut($G$,$m$);
3: for $k \leftarrow 1$ to $m$ do
4:     ID $\leftarrow -1$;
5:     for each node $w \in g_k$ do
6:         if $w$ is marked then
7:             ID $\leftarrow w.mark$;
8:             break;
9:         end if
10:    end for
11:    for each node $w \in g_k$ do
12:        if $w \in \{v_1, v_2, ..., v_n\}$ then
13:            $\phi(w) \leftarrow p_{ID}$;
14:        end if
15:    end for
16: end for
17: return $\phi$
```

Algorithm 2 CreateGraph

```
Input: $V$={$v_1$,$v_2$,...,$v_n$}:new VMs
    $E$={$e_1$,$e_2$,...,$e_Y$}:existing VMs
    $P$={$p_1$,$p_2$,...,$p_m$}:physical servers
    $S_i$:Number of available slots on $p_i$
    $T_i$:Number of relevant existing VMs on $p_i$
    $M$:$M_{ij}$ is the traffic from $v_i$ to $v_j$
    $F$:$F_{ij}$ is the traffic between $v_i$ and $e_j$
Output: $G$(The graph to be partitioned)
1: $G \leftarrow$InitializeGraph($V$,$E$,$M$,$F$);
2: for each existing VM $e_j \in E$ do
3:     $e_j$.mark $\leftarrow \pi(e_j)$.ID;
4: end for
5: $G \leftarrow$AddDummyVMs($G$,$\sum_{i=1}^{m} S_i - |V|$);
6: $C_{max} \leftarrow max\{S_i + T_i\}, 1 \leq i \leq m$;
7: $S_{max} \leftarrow max\{S_i\}, 1 \leq i \leq m$;
8: $\delta \leftarrow 2S_{max} - C_{max} + 1$;
9: for $i \leftarrow 1$ to $m$ do
10:    $B_i \leftarrow C_{max} - S_i$;
11:    if $\delta > 0$ then
12:        $B_i \leftarrow B_i + \delta$;
13:    end if
14:    $G \leftarrow$ AddPseudoVMs($G$, $B_i - T_i, i$);
15:    $G \leftarrow$ ConnectExistingVMs($G, i, \infty$,"tree");
16: end for
17: return $G$
```

(line 2), while minimizing the weight of edges connecting different subgraphs. Third, each subgraph is mapped to a specific physical server, by the mark of existing VMs within it. Since our mechanisms ensure that only one type of existing VMs exist in $g_k(1 \leq k \leq m)$, by getting the mark of them (ID in line 7), we map g_k to physical server p_{ID}. In this way, the total amount of traffic brought by new VMs is minimized. The computing complexity of DVMP is dominated by BalancedMinKCut(.), which is adapted from [14]. The time complexity of BalancedMinKCut(.) is $O(|G|^4)$, and its approximation ratio is $\frac{m-1}{m}|G|$.

Alg. 2 shows the creation of graph G. First, G is initialized as a graph of new VMs and existing VMs (line 1). Existing VMs are marked with their hosting server identities to indicate that they should be mapped back to their original servers (line 2-4). Second, $\sum_{i=1}^{m} S_i - n$ dummy VMs are added to G as isolated nodes (line 5). Third, we unify the number of slots on each server to C_{max} by adding pseudo slots, then apply a change (*i.e.,* δ, if $\delta > 0$) on the number of pseudo slots to deal with the third challenge (line 6-17). For server $p_i(1 \leq i \leq m)$, AddPseudoVMs(.) is invoked to add $B_i - T_i$ pseudo slots, which are occupied by $B_i - T_i$ pseudo VMs (marked with i). Pseudo VMs are regarded as existing VMs for server $p_i(1 \leq i \leq m)$, and ConnectExistingVMs(.) adds pseudo edges with weight of ∞ to connect them with existing VMs on p_i: $\{e_j|\pi(e_j) = p_i, 1 \leq j \leq Y\}$. Fig. 6 is an example of Alg. 2's output graph G, in which new VMs, existing VMs, dummy VMs and the pseudo VMs are the nodes. The pseudo edges, communication relationship among new VMs, communication relationship between new VMs and existing VMs all together form the edges of new graph G'.

IV. Evaluation

A. Simulation Setup

We compare DVMP with the following VM placement schemes.

- **Random Placement:** VMs are tightly placed into the available physical servers, without considering the network traffic among them, as existing capacity planning tools have done, *e.g.,* VMware Capacity Planner [18] and IBM WebSphere CloudBurst [19].
- **Greedy Placement:** VMs are tightly placed into the available physical servers in a greedy way, *i.e.,* VMs with higher traffic demands are put into the same server with higher priority.
- **Naive BMKP Placement:** VMs are placed based on the naive BMKP model (prior traffic-aware VM placement [10], [11]), without considering the communication traffic with existing VMs. When the available VM slots are unequal in physical servers, our mechanisms are used to translate the problem, as discussed in Section III.

For the traffic among VMs, we use the same traffic pattern as [11], which is collected from a private data center testbed [20]. The communication among VMs is sparse and

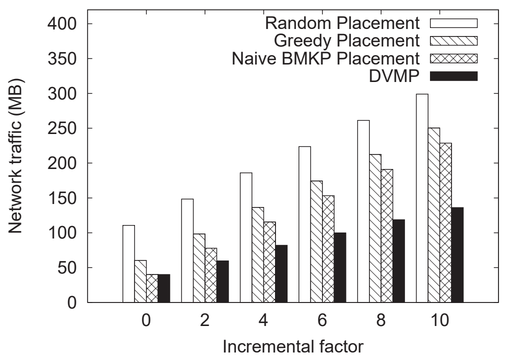

Fig. 7. Network traffic of the four schemes against size of communication traffic with existing VMs.

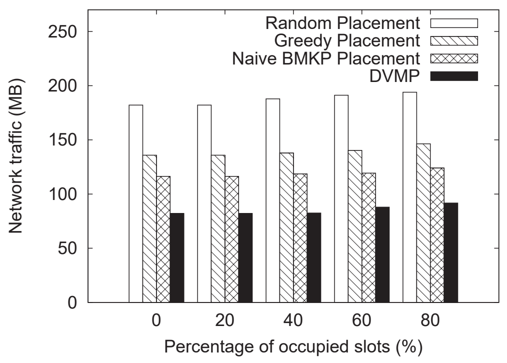

Fig. 9. Network traffic of the four schemes against percentage of occupied slots.

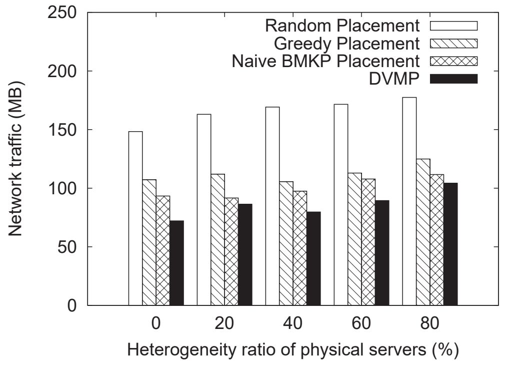

Fig. 8. Network traffic of the four schemes against server heterogeneity ratio.

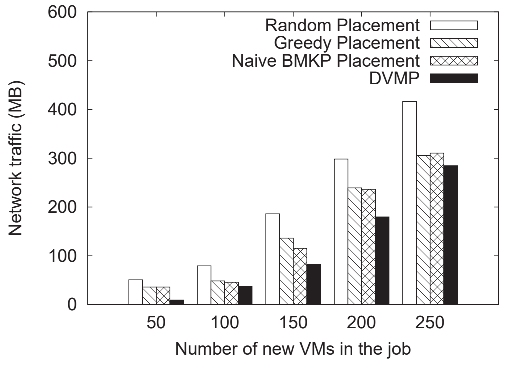

Fig. 10. Network traffic of the four schemes against number of new VMs.

skewed, *i.e.,* the majority VMs have traffic size of 10∼100 KB, while about 4% VMs have 1.5∼2.5 MB. At any time, only 15∼20% VMs communicate with each other, *i.e.,* at most 4% of all to all communication pairs. Considering the sparsity of inter VM communication, we set the percentage of communicating VMs to be 4% for each VM. Unless otherwise specified, the new job has 150 new VMs waiting to be placed on 64 physical servers. Due to server heterogeneity, the number of VM slots (C_i) for each server ranges from 24 to 32. For server p_i, the number of relevant existing VMs varies in $[0, \frac{1}{4}C_i], 1 \leq i \leq m$.

B. Impact of Traffic Size with Existing VMs

According to the survey in section II-B, traffic with existing VMs is on average 4 times the size of traffic among new VMs. We test the impact of traffic with existing VMs on the four VM placement schemes, and the traffic size with existing VMs is set as 0, 2, 4, 6, 8 and 10 times that among new VMs, respectively. For simplicity, we call this factor the *incremental factor*. When the incremental factor is 0, it means there is no traffic with existing VMs, and larger incremental factor means more communication traffic with existing VMs. The network traffic of the four VM placement schemes with different incremental factors is shown in Fig. 7.

We can find that the network traffic caused by new VMs increases with more communication traffic with existing VMs. When the incremental factor is 0 (*i.e.,* there is no communication with existing VMs), DVMP acts the same way as naive BMKP. However, with the increase of incremental factor, DVMP saves more and more network traffic compared with other schemes, because it considers both types of traffic and optimizes jointly. Greedy placement and naive BMKP placement cause less traffic compared with random placement, since they are traffic-aware and place VMs with large mutual communication demand on the same physical server.

C. Impact of Server Heterogeneity

We then explore the impact of server heterogeneity on the four VM placement schemes. We denote the maximum and minimum number of VM slots of servers as C_{max} and C_{min}, then define the server *heterogeneity ratio* as $r = \frac{C_{max}-C_{min}}{C_{max}}$. Hence, the number of VM slots for each physical server ranges from $(1-r)C_{max}$ to C_{max}. Larger heterogeneity ratio indicates more imbalanced physical servers. We compare the network traffic of the four VM placement schemes with r being 0%, 20%, 40%, 60% and 80%, respectively, as shown in Fig. 8. The result tells that the network traffic slightly increases with higher server heterogeneity ratio. The reason is as follow. Since the minimum number of slots for each physical server is $C_{min} = (1-r)C_{max}$, servers will possibly have fewer slots with higher r. Hence, more physical servers are needed to place VMs, and the network traffic across servers

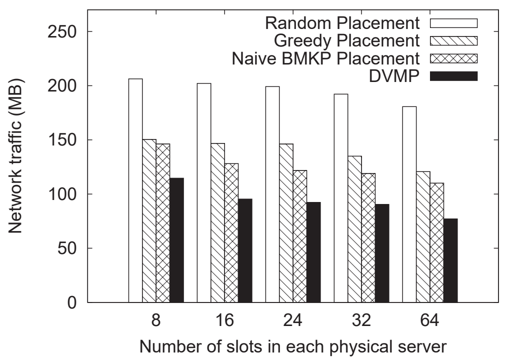

Fig. 11. Network traffic of the four schemes against number of slots per physical server.

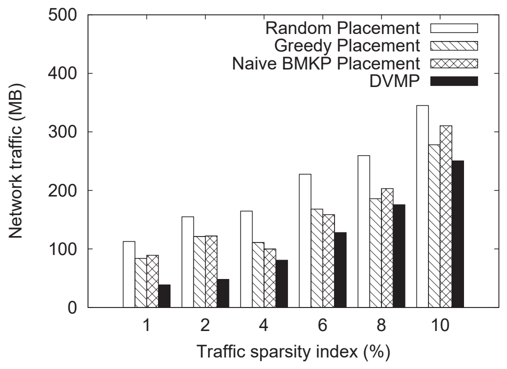

Fig. 13. Network traffic of the four schemes against traffic sparsity.

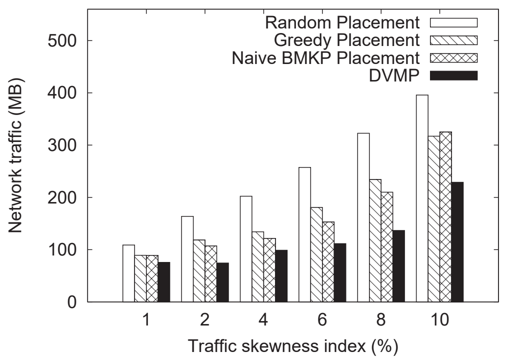

Fig. 12. Network traffic of the four schemes against traffic skewness.

increases accordingly. When $r = 0\%$, physical servers are homogeneous, but the problem is still imbalanced due to previously deployed VMs. Besides, it can be concluded from Fig. 8 that DVMP always saves the most network traffic under various server heterogeneity conditions.

D. Impact of Percentage of Occupied Slots

As analyzed previously, server imbalance is caused by both server heterogeneity and previously deployed VMs. Next we explore the impact of previously deployed VMs on the placement performance and compare the four placement schemes with different numbers of existing VMs. The percentage of occupied slots on each physical server is set to 0%, 20%, 40%, 60% and 80%, respectively. Results are shown in Fig. 9.

Network traffic of random placement slightly rises when more slots are occupied by existing VMs. It is because the number of available slots on each physical server reduces when more slots are occupied, thus more physical servers are needed to accommodate new VMs, resulting more traffic across servers. On the other hand, greedy placement, naive BMKP placement and DVMP are barely influenced, due to the consideration of network traffic and VMs with high mutual traffic demand are placed on the same server with high priority.

E. Impact of the Number of New VMs

In the incremental VM placement scenario, some VMs of the tenant have been deployed previously, and n new VMs are requested to expand the computing capacity. We validate the effectiveness of DVMP in reducing overall network traffic with different numbers of new VMs requested, *i.e.,* n = 50, 100, 150, 200 and 250, respectively. Then we place these new VMs according to the four schemes, and calculate the network traffic caused by new VMs, as shown in Fig. 10.

It is observed that DVMP outperforms state-of-the-art VM placement schemes, since it considers both types of traffic and maximally localizes overall network traffic. Random placement scheme poses the most traffic to data center networks, because it is traffic-oblivious and simply consolidates VMs onto as fewer physical servers as possible. Greedy placement and naive BMKP placement are traffic-aware, but fail to consider the communication traffic with existing VMs, hence may generate suboptimal solutions. With more new VMs to place, network traffic caused by the four VM placement schemes increases, as the total traffic demand increases while only a part of it is localized.

F. Impact of the Number of VM Slots on Servers

Due to server heterogeneity, the maximum number of VMs that each server can accommodate may be different. With the evolvement of hardware technologies, data center servers are more powerful, and it's common practice to simultaneously run tens of VMs on a single physical server [21]. Denoting C_{max} as the maximum number of VMs a server can accommodate, we show the impact of server hardware profiles on VM placement performance, *i.e.,* C_{max}=8, 16, 24, 32 and 64, respectively.

Intuitively, with more powerful physical servers (*i.e.,* more VM slots on each server), there is more room for VMs with high traffic demand to coexist on the same server to save traffic. Simulation results demonstrate this idea, as shown in Fig. 11, network traffic decreases with the increase of C_{max}. For instance, the traffic of DVMP decreases by 32.78% when C_{max} increases from 8 to 64. Random placement benefits the least due to its random nature, high traffic demand among VMs may not be localized.

G. Impact of Traffic Skewness

Data center network traffic is quite skewed and sparse [10], [11], [22]. The traffic demand for the majority of VMs is roughly the same, while a small portion of VMs have very large traffic demand. We denote the *skewness index* as the percentage of VMs with large traffic demand. According to the survey on Bing data centers [22], 0.1% of VMs generate 60% of all traffic and 4.8% of them causes 99% of overall traffic. Similar results are observed in [10], [11], and the *skewness index* is about 4%. In order to test DVMP's performance in the general situations, we compare the four VM placement schemes with different *skewness indexes*, *i.e.,* 1%, 2%, 4%, 6%, 8% and 10%, respectively.

Fig. 12 shows the corresponding simulation results. First, when *skewness index* increases, there are more overall network traffic among VMs, and the resulting network traffic across physical servers is much more. Second, the gap between DVMP and the other schemes grows with higher *skew index*. Given the practice of 4% skewness index, in real data centers DVMP can save 48.2%, 22.4% and 19.6% network traffic compared with random placement, greedy placement and naive BMKP placement, respectively.

H. Impact of Traffic Sparsity

In practical data centers, traffic among VMs is quite sparse, and not each VM communicates with every one else [11]. At any time, there are only about 15% to 20% of VMs communicating with each other. We denote traffic *sparsity index* as the percentage of communicating VMs. With lower sparsity index, the communication among VMs is sparser. We compare the four VM placement schemes with different sparsity index, as shown in Fig. 13.

We observe that the network traffic increases for all VM placement schemes, while DVMP causes the least network traffic all the time. Besides, the gap between DVMP and random placement reduces with more inter VM communications, since there is less room for traffic-aware VM placement schemes to localize network traffic. According to [11], the sparsity index in practical data centers is about 4%, in which case DVMP can save 54.2%, 21.5% and 18.3% traffic compared with random placement, greedy placement and naive BMKP placement, respectively.

V. Related Work

Many prior VM placement works [23], [24], [25], [26], [27] focus on consolidating VMs on physical servers to improve resource utilization (*e.g.,* CPU, memory, disks), while at the same time satisfying constraints of server resources. In these works, VM placement is usually modeled as a constrained optimization problem, which targets at minimizing the number of physical servers used. Actually, some commercial softwares are currently available for deciding the placement location of new VMs, *e.g.,* VMware Capacity Planner [18] and IBM WebSphere CloudBurst [19]. Since physical server resources include not only CPU, memory and disks, but also the network bandwidth, which is neglected by the previously mentioned traffic-oblivious works. When VM pairs with high mutual traffic demand are unfortunately placed on remote physical servers, the consequent communication may result in great burden for the network infrastructure.

Motivated by the drawback of traffic-oblivious works, traffic-aware VM placement schemes are proposed to reduce the traffic burden of networks. Wang *et al.* [28] minimize the number of servers with dynamic bandwidth demand when placing VMs, they formulate the VM placement as a Stochastic Bin Packing problem and propose an online packing algorithm. Meng *et al.* [10] improve the scalability of data center networks with traffic-aware VM placement, by making the traffic pattern among VMs better aligned with communication distance between them, *i.e.,* VMs with large mutual bandwidth are assigned to close hosting machines. A key phase of their algorithm is the balanced minimum k-cut, which minimizes the amount of network traffic posed to the network. Compared to DVMP, they don't consider the traffic interaction with existing VMs, and thus may generates suboptimal solutions in practice.

Fang *et al.* [11] propose VMPlanner, a framework for reducing power costs of network elements in data centers. The basic idea is reducing traffic posed to the network by balanced minimum k-cut, then greedy bin packing is applied to consolidate flows into as fewer switches and links as possible to save network power. They assume physical servers are homogeneous, which may be impractical in practice. Besides, the traffic interaction with existing VMs is also neglected. Alicherry *et al.* [16] reduce the data access latency for data intensive cloud applications in data centers. Given the location of data sets, they place new VMs with optimization goal of minimizing data access latency while satisfying server capacity constraints. This is similar to the placement of mappers and reducers in a data-local manner. When it's infeasible to place VMs exactly on the same server with data, servers close to them are preferentially chosen. As demonstrated previously, this greedy placement may also lead to suboptimal solution, as a result of neglecting inter-VM communication.

Jiang *et al.* [29] exploit joint routing and VM placement to optimize their network performance (*e.g.,* bisection bandwidth, throughput) and cost (*e.g.,* power consumption). By leveraging and expanding Markov approximation technique, a limited number of VM migrations is needed to achieve their goals. Cohen *et al.* [30] focus on the bandwidth-constrained VM placement problem, and the goal is to maximize the benefit from the overall communication sent by the VMs to a single designated point in the data center when considering a storage area network of applications with intense storage requirements. Their polynomial time constant approximation algorithm show great performance in the simulation using traces from real production data centers.

Guo *et al.* [31] propose a shadow routing based VM placement scheme in large heterogeneous data centers or server clusters and show the good performance, robustness and adaptability of it. By appropriately dealing with VM-to-PM packing constraints, they produce an asymptotically optimal solution and show the computational feasibility in practical implementations. Dong *et al.* [32] focus on energy saving VM placement problem of optimizing both physical server resources utilization and network resources utilization simultaneously. They propose a VM placement scheme that meets multiple resource constraints, including physical server side(CPU, memory, storage, bandwidth, *etc.*) and network link capacity. Wen *et al.* [33] explore the opportunity to address the continuous congestion via optimizing VM placement in virtualized datacenters and propose VirtualKnotter, an efficient online VM placement algorithm to reduce congestion with controllable VM migration traffic as well as low time complexity.

VI. Conclusion

This paper presents a novel method (DVMP) for reducing the traffic amount posed to the network by intelligent VM placement in virtualized data centers. With optimized placement scheme, both communication traffic among new VMs and that with existing VMs are maximally localized. Server imbalance due to hardware heterogeneity and partial deployment are also considered. We make a number of mechanisms and algorithms to translate the new problem to a variant of BMKP problem and develop an efficient algorithm to get the solution. Evaluation results show that DVMP indeed saves considerable network traffic compared with state-of-the-art VM placement schemes.

References

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

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1–10, IEEE, 2010.

[3] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: a high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, (New York, NY, USA), pp. 63–74, ACM, 2009.

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*, SIGCOMM '08, (New York, NY, USA), pp. 63–74, ACM, 2008.

[5] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, (New York, NY, USA), pp. 51–62, ACM, 2009.

[6] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "Ficonn: Using backup port for server interconnection in data centers," in *INFOCOM 2009, IEEE*, pp. 2276–2285, 2009.

[7] P. Costa, A. Donnelly, A. Rowstron, and G. OShea, "Camdoop: Exploiting in-network aggregation for big data applications," in *USENIX NSDI*, vol. 12, 2012.

[8] H. Xie, G. Shi, and P. Wang, "Tecc: Towards collaborative in-network caching guided by traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*, pp. 2546–2550, 2012.

[9] B. Fitzpatrick, "Distributed caching with memcached," *Linux journal*, vol. 2004, no. 124, p. 5, 2004.

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

[11] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179 – 196, 2013.

[12] D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," in *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pp. 256–265, 2011.

[13] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 17–17, USENIX Association, 2012.

[14] H. Saran and V. V. Vazirani, "Finding k cuts within twice the optimal," *SIAM Journal on Computing*, vol. 24, no. 1, pp. 101–108, 1995.

[15] K. Andreev and H. Racke, "Balanced graph partitioning," *Theory of Computing Systems*, vol. 39, no. 6, pp. 929–939, 2006.

[16] M. Alicherry and T. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *INFOCOM, 2013 Proceedings IEEE*, pp. 647–655, IEEE, 2013.

[17] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 68–73, Dec. 2008.

[18] V. C. Planner, "http://www.vmware.com/products/capacityplanner/."

[19] I. W. CloudBurst, "http://www-01.ibm.com/software/webservers/cloudburst/."

[20] L. Feng, Z. Baopeng, M. Kai, H. Jackson, W. Xinran, and M. Wenbo, "A cloud test bed for china railway enterprise data center," *Technical report, Intel Corporation*, 2009.

[21] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer.," in *Hotnets*, 2009.

[22] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 431–442, ACM, 2012.

[23] A. Kochut, "On impact of dynamic virtual machine reallocation on data center efficiency," in *Modeling, Analysis and Simulation of Computers and Telecommunication Systems, 2008. MASCOTS 2008. IEEE International Symposium on*, pp. 1–8, 2008.

[24] W. Leinberger, G. Karypis, and V. Kumar, "Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints," in *Parallel Processing, 1999. Proceedings. 1999 International Conference on*, pp. 404–412, IEEE, 1999.

[25] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM'07. 10th IFIP/IEEE International Symposium on*, pp. 119–128, IEEE, 2007.

[26] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Int. CMG Conference*, pp. 399–406, 2007.

[27] S. Mehta and A. Neogi, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 363–370, IEEE, 2008.

[28] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM, 2011 Proceedings IEEE*, pp. 71–75, IEEE, 2011.

[29] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *INFOCOM, 2012 Proceedings IEEE*, pp. 2876–2880, IEEE, 2012.

[30] R. Cohen, L. Lewin-Eytan, J. Seffi Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *INFOCOM, 2013 Proceedings IEEE*, pp. 355–359, IEEE, 2013.

[31] Y. Guo, A. Stolyar, and A. Walid, "Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud," in *INFOCOM, 2013 Proceedings IEEE*, pp. 620–628, 2013.

[32] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pp. 618–624, IEEE, 2013.

[33] X. Wen, K. Chen, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pp. 12–21, IEEE, 2012.