Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement

Xiaoqiao Meng, Vasileios Pappas, Li Zhang
IBM T.J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
Email: {xmeng, vpappas, zhangli}@us.ibm.com

***Abstract*—The scalability of modern data centers has become a practical concern and has attracted significant attention in recent years. In contrast to existing solutions that require changes in the network architecture and the routing protocols, this paper proposes using traffic-aware virtual machine (VM) placement to improve the network scalability. By optimizing the placement of VMs on host machines, traffic patterns among VMs can be better aligned with the communication distance between them, e.g. VMs with large mutual bandwidth usage are assigned to host machines in close proximity. We formulate the VM placement as an optimization problem and prove its hardness. We design a two-tier approximate algorithm that efficiently solves the VM placement problem for very large problem sizes. Given the significant difference in the traffic patterns seen in current data centers and the structural differences of the recently proposed data center architectures, we further conduct a comparative analysis on the impact of the traffic patterns and the network architectures on the potential performance gain of traffic-aware VM placement. We use traffic traces collected from production data centers to evaluate our proposed VM placement algorithm, and we show a significant performance improvement compared to existing generic methods that do not take advantage of traffic patterns and data center network characteristics.**

I. Introduction

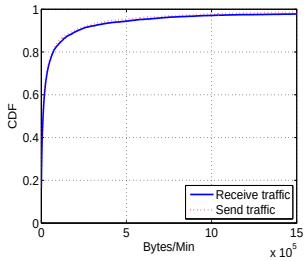
Modern virtualization based data centers are becoming the hosting platform for a wide spectrum of composite applications. With an increasing trend towards more communication intensive applications in data centers, the bandwidth usage between virtual machines (VMs) is rapidly growing. This raises a number of concerns with respect to the scalability of the underlying network architecture, an issue that has attracted significant attention recently [1][2][3][4][5][6][7]. Techniques in these proposals include rich connectivity at the edge of the network and dynamic routing protocols to balance traffic load.

In this paper, we tackle the scalability issue from a different perspective, by optimizing the placement of VMs on host machines. Normally VM placement is decided by various capacity planning tools such as VMware Capacity Planner [8], IBM WebSphere CloudBurst [9], Novell PlateSpin Recon [10] and Lanamark Suite [11]. These tools seek to consolidate VMs for CPU, physical memory and power consumption savings, yet without considering consumption of network resources. As a result, this can lead to situations in which VM pairs with heavy traffic among them are placed on host machines with large network cost between them. To understand how often this happens in practice, we conducted a measurement study in operational data centers and observed three apparent trends: there is a low correlation between the average pairwise traffic rate and the end-to-end cost; traffic distribution for individual VMs is highly uneven; VM pairs with relatively heavier traffic rate tend to constantly exhibit the higher rate and conversely VM pairs with low traffic rate tend to exhibit the low rate. These three observations suggest that there is a great potential in optimizing VM placement to save bandwidth and realizing such potential is feasible.

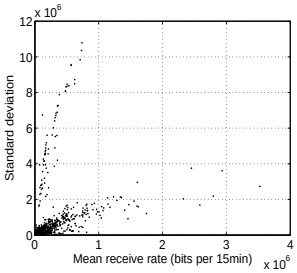
We formally define the Traffic-aware VM Placement Problem (TVMPP) as an optimization problem. Input to such a problem includes the traffic matrix among VMs and the cost matrix among host machines. The optimal solution to the TVMPP dictates where VMs should be placed in order to minimize the objective, which, with a typical cost matrix definition, is the aggregate traffic rates perceived by every switch. We prove that the TVMPP is NP-hard and propose a heuristic algorithm to solve the TVMPP efficiently even for large problem sizes. The proposed algorithm takes a novel two-tier approach: it first partitions VMs and hosts into clusters separately, then it matches VMs and hosts at cluster level and consequently at individual level. We evaluate the algorithm by using traces collected from production data centers. The experimental results show that the proposed algorithm can significantly lower the aggregate traffic and reduce the computational time when compared against two existing representative methods.

Next, given the variety of the proposed network architectures and the complexity of traffic patterns in data centers, we investigate under which conditions it is beneficial to apply the VM placement approach in practice. Specifically, we consider four architectures under different traffic patterns. We seek to answer the following question: which network architecture and what traffic pattern yields the highest performance gain when the TVMPP is optimally solved? Our study shows that design choices in each proposed architecture have profound impact on the network scalability improvement achieved by the TVMPP. While multi-level network architectures, such as BCube [5], reap the most benefit, architectures that have been designed with uniform traffic patterns in mind, such as VL2 [3], see little benefit in optimally solving the TVMPP. On the other hand, non-uniform traffic patterns, either due to the variability in VM usage or due to localized traffic clusters among VMs, create higher benefit for optimally solving the TVMPP.

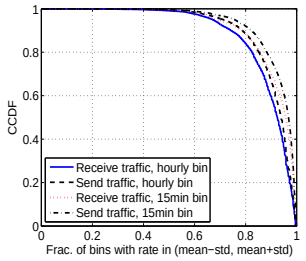
The main contributions of this paper are summarized as



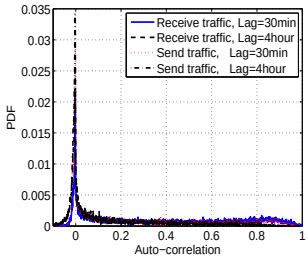
(a) CDF of mean traffic rate



(b) Distribution of (mean, standard deviation)



(c) CCDF of proportions of stable intervals



(d) PDF of auto-correlation coefficients

Fig. 1. Statistics about VM traffic in 17,000 VMs dataset

following:

- We address the scalability issue of data center networks with network-aware VM placement. We formulate it as an optimization problem, prove its hardness and propose a novel two-tier algorithm.
- We analyze the impact of data center network architectures and traffic patterns on the scalability gains attained by network-aware VM placement.
- We measure traffic patterns in production data center environments, and use the data to evaluate the proposed algorithm as well as the impact analysis.

II. Background and Motivation

In this section, we present a few observations on the traffic patterns in production data centers. We also describe four data center network architectures, including the traditional tree-like architecture and three recently proposed one.

A. Data Center Traffic Patterns

To better understand data center traffic patterns, we examine traces from two data-center-like systems. The first comes from a data warehouse hosted by IBM Global Services. It collects server resource utilization information from hundreds of server farms. Each server farm contains physical hosts and VMs that are used by various enterprise customers. Our study is focused on the incoming and outgoing traffic rates for 17 thousand VMs. The second trace is from a server cluster with about hundreds of VMs. Due to some practical constraints, we are able to measure the incoming and outgoing TCP connections for 68 VMs. We also measure the round-trip latency between

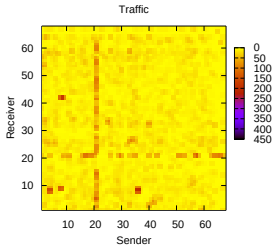


Fig. 2. Traffic matrix

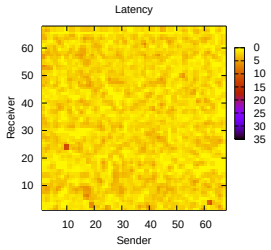


Fig. 3. Latency matrix

any two servers. This measurement continues for 10 days. Based on these two traces, we observe the following three trends regarding traffic patterns in data centers.

Uneven distribution of traffic volumes from VMs: Figure 1.(a) is the CDF for the outgoing and incoming aggregate traffic rate at individual VMs in the warehouse dataset. The traffic rate is the average over a two-week period. While 80% of VMs have average rate less than 800 KBytes/min, 4% of them have a rate ten times higher. Since the original data do not report throughput between VMs, we use the second trace which measures TCP-connection-per-hour between every two VMs. Figure 2 uses different levels of grayscale to reflect rates. The inter-VM traffic rate indeed varies significantly. Besides the VM 21 which has clearly high traffic volume with almost every other VM, several other VM pairs also exhibit high data rates.

Stable per-VM traffic at large timescale: Although the average traffic rates are divergent among VMs, the rate for a large proportion of VMs are found to be relatively stable when the rate is computed at large time intervals. Such an observation is apparent in Figures 1.(b), 1.(c) and 1.(d). In Figure 1.(b) we plot the mean and standard deviation of traffic rates for 17K VMs. It shows that for the majority of VMs (82%), the standard deviation of their traffic rates is no more than two times of the mean. Figure 1.(c) is the CCDF for the percentage of *stable* time intervals. An interval is labeled as stable if the rate within it is no more than one standard deviation away from the mean in the entire measurement period. Figure 1.(c) shows that for 82%-92% of VMs, no less than 80% of the entire set of time intervals are stable. We also examine traffic constancy by computing auto-correlation coefficients from the traffic timeseries for each VM. Figure 1.(d) shows the PDF for the auto-correlation coefficients at two lags: 30 minutes and 4 hours. The long tail indicates that at the two large timescales, a large fraction of the VMs' traffic rates are relatively constant.

Weak correlation between traffic rate and latency: Based on our measurement on the traffic rate and end-to-end latency among 68 VMs in a production cluster, in Figure 2 we show the pairwise traffic intensity, and in Figure 3 we show the pairwise end-to-end latency. While in Figure 2, darker color indicates higher traffic rate; on Figure 3, darker color indicates lower latency. An apparent observation is that the VM pairs

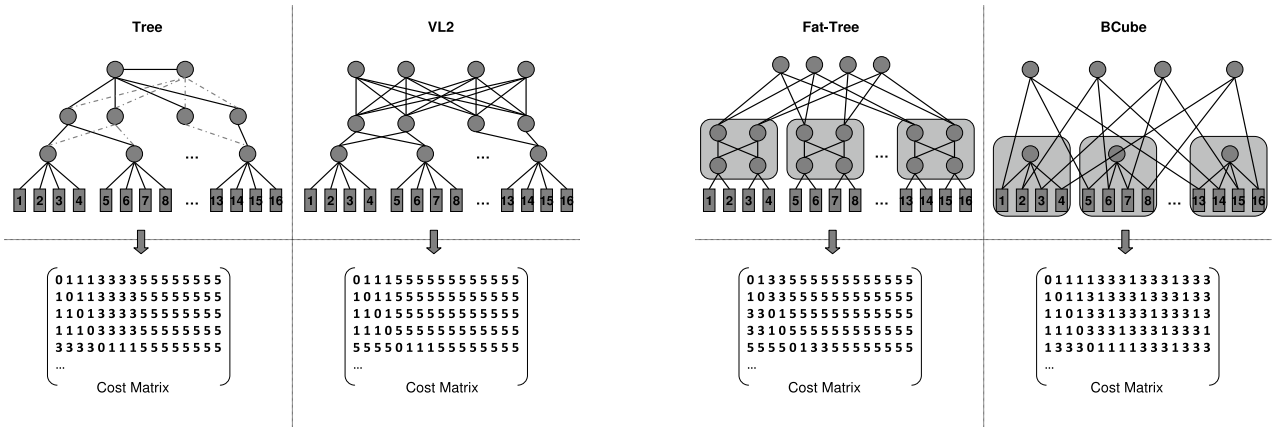


Fig. 4. Network topologies and corresponding cost matrices for four data center network architectures

with high rate do not necessarily correspond to low latency and vice versa. The correlation coefficient for these two matrices is -0.32, indicating a fairly weak correlation.

The above observations make a case for shuffling VM placement such that more traffic are localized and the pairwise traffic demand is better coordinated with the pairwise network cost. The potential benefit is two-fold: increased network scalability and reduced average traffic latency. In addition, the observed traffic stability over large timescale suggests that it is feasible to find good placements based on past traffic statistics. Finally, the observed large time scales of stable traffic suggest an amortized VM migration cost.

B. Data Center Network Architectures

Current data centers follow to a great extend a common network architecture, known as the three-tier architecture [12]. At the bottom level, known as the access tier, each server connects to one (or two, for redundancy purposes) access switch. Each access switch connects to one (or two) switches at the aggregation tier, and finally, each aggregation switch connects with multiple switches at the core tier. While the physical topology in such three-tier architecture is a multi-rooted forest topology, in reality packets are forwarded according to the logical layer-2 topology that is created with the use of VLANs and the spanning tree algorithm. This layer-2 topology is always a tree, usually rooted at one of the core switches. Figure 4 shows one such topology with 16 servers and one VLAN (labeled as *Tree*).

Scaling the three-tier architecture is achieved by scaling up each individual switch, i.e. by increasing its fan-out, rather than scaling out the topology itself. For example, based on [12], the core tier can accommodate 8 switches at most. Topology scaling limitations as well as other ones such as the need for flat address space, or the high server over-subscription, have prompted recently many parallel efforts in redefining the network architecture of the data centers. Next, we present three alternative architectures that have been proposed in the last two years.

VL2 [3] is a new architecture that shares many features with the previous one. More specifically, it is a 3-tier architecture with main difference that the core tier and the aggregation tier form a Clos [13] topology, i.e. the aggregation switches are connected with the core ones by forming a complete bipartite graph. In addition, traffic originated from the access switches is forwarded in the aggregation and the core tiers with the use of valiant load balancing, i.e. it is forwarded first to a randomly selected core switch and then back to the actual destination. The main rationale behind this architecture is that when traffic is unpredictable the best way to balance load across all available links is to randomly select a core switch as an intermediate destination. Figure 4 shows one such network with 16 servers (labeled as *VL2*).

PortLand [2] is another three-tier architecture that shares with the VL2 the same Clos topology feature, though at different levels. The PortLand architecture makes use of fat-tree [14] topologies and it is built around the concept of pods: a collection of access and aggregation switches that form a complete bipartite graph, i.e., a Clos graph. In addition, each pod is connected with all core switches, by evenly distributing the up-links between all the aggregation switches of the pod. As such, a second Clos topology is generated between the core switches and the pods. PortLand assumes all switches are identical, i.e., they have the same number of ports (something not required by the previous two architectures). The number of available ports per switch is the only parameter that determines the total number of pods and in consequence the total number of required switches as well as connected servers. More specifically, if k is the number of ports on each switch, then there are k pods, with $\frac{k}{2}$ access switches and $\frac{k}{2}$ aggregation switches in each pod. Each pod is connected with $\frac{k^2}{4}$ core switches and with $\frac{k^2}{4}$ servers. Thus in total, there are $\frac{5k^2}{4}$ switches that interconnect $\frac{k^3}{4}$ servers. Figure 4 shows one such network for $k = 4$ (labeled as *Fat-tree*).

BCube [5] is a new multi-level network architecture for the data center with the following distinguishing feature: servers are part of the network infrastructure, i.e., they forward packets on behalf of other servers. BCube is a recursively defined structure. At level 0, BCube_0 consists of n servers that

connect together with a n-port switch. A BCube$_k$ consists of n BCube$_{k-1}$ connected with n^k n-port switches. Servers are labeled based on their locations in the BCube structure. E.g., in a three-layer BCube, if a server is the third server in a BCube$_0$ that is inside the second BCube$_1$ being inside the fourth BCube$_2$, then its label is *4.2.3*. Based on such labels servers identify with which they connect to: servers whose label differs only at the ith level connect together with a switch at the BCube$_i$ level. In essence, BCube is a generalized hypercube architecture [15] with the main difference that neighboring nodes instead of forming a full mesh with each other, they connect through switches. Note that the number of connected servers as well as the number of required switches in a BCube is a function of n, the total port number of each switch, and k, the number of BCube levels. Figure 4 shows one such network for $k = 1$ and $n = 4$ (labeled as *BCube*).

These four architectures have been designed independently with different goals in mind. In this paper we aim to understand how these topologies compare against each other when traffic loads can be moved around with the use of VMs.

III. Virtual Machine Placement Problem

In this section we formally define the VM placement problem and analyze its complexity.

A. Problem Formulation

We study the problem of placing VMs on a set of physical hosts (hereinafter referred to as hosts). We assume existing CPU/memory based capacity tools have decided the number of VMs that a host can accommodate. Thus we use a *slot* to refer to one CPU/memory allocation on a host. Multiple slots can reside on the same host and each slot can be occupied by any VM. We consider a scenario where there are n VMs and n slots. By assuming static and single-path routing which are the current typical settings, we use C_{ij}, a fixed value, to refer to the communication cost from slot i to j. D_{ij} denotes traffic rate from VM i to j. e_i denotes external traffic rate for VM i. Without loss of generality, we assume all external traffic are routed through a common gateway switch. Thus we can use g_i to denote the communication cost between VM i and the gateway. For any placement scheme that assigns n VMs to n slots on a one-to-one basis, there is a corresponding permutation function $\pi : [1, \ldots, n] \rightarrow [1, \ldots, n]$. We can formally define the Traffic-aware VM Placement Problem (TVMPP) as finding a π to minimize the following objective function

$$\sum_{i,j=1,\ldots,n} D_{ij} C_{\pi(i)\pi(j)} + \sum_{i=1,\ldots,n} e_i g_{\pi(i)} \quad (1)$$

The above objective function is equivalent to

$$\min_{X \in \Pi} tr(DX^T C^T X) + eX^T g^T \quad (2)$$

where $tr()$ is defined as $tr(A) = \sum_i A_{ii}$ for an input matrix A. X is a permutation matrix which must satisfies three constraints: $X_{ij} \in \{0, 1\} (\forall i, j), \sum_{j=1}^{n} X_{ij} = 1 (\forall i), \sum_{i=1}^{n} X_{ij} = 1 (\forall j)$. Π is the set of all valid permutation matrices. e, g are row vectors. In the above formulation, except for π, all the other variables are assumed to be known.

The meaning of the objective function in (1) depends on the definition of C_{ij}. In fact C_{ij} can be defined in many ways. For the sake of illustration, we define C_{ij} as the number of switches on the routing path from VM i to j. With such a definition, the objective function is the sum of the traffic rate perceived by every switch. If the objective function is normalized by the sum of VM-to-VM bandwidth demand, it is equivalent to the average number of switches that a data unit traverses. If we further assume every switch causes equal delay, the objective function can be interpreted as the average latency for a data unit traversing the network. Accordingly, optimizing TVMPP is equivalent to minimizing average traffic latency caused by network infrastructure.

The formulation described above assumes equal number of VMs and slots. When there are more available slots than VMs, we can always make the two numbers equal by introducing dummy VMs which do not receive or send any traffic. Obviously adding such dummy VMs does not affect VM placement. Notice that the second part in the objective function of (1) is the total external traffic rate calculated at all switches. In reality, this sum is most likely constant regardless of VM placement, because in typical data center networks, including those in Figure II-A, the cost between every end host and the gateway is the same. Therefore, the second part in the objective function can be ignored in our analysis.

The TVMPP framework is very general and can be applied in both offline and online scenarios. In a typical offline scenario, multiple customers request for VMs. The data center operators first estimate the traffic matrix based on customers' input, then collect the network topology information, and solve the TVMPP problem to decide which host(s) should be used to create VMs. In a typical online scenario, VMs have been created and running, the operators periodically collect traffic matrix and re-solve the TVMPP problem. If a new VM placement scheme is found to yield better objective value, the operators will decide whether a reshuffling of the VMs assignment is needed.

B. Complexity Analysis

We now analyze the computational complexity of TVMPP. When C and D are matrices with arbitrary real values, TVMPP falls into the category of *Quadratic Assignment Problem (QAP)* in the Koopmans-Beckmann form [16]. QAP is a known NP-hard [17] problem. In fact, it is one of the most difficult problems in NP-hard class - as shown in [17], even finding an σ-approximation algorithm (σ is a constant) is NP-hard. Although various methods for solving QAP have been proposed [16], there is a general agreement that finding the optimality of QAP problems with size > 15 is practically impossible.

It is conceivable that the TVMPP is a special case of QAP since the data center network architecture imposes special constraints on D. We further prove the following:

Theorem 1: For a TVMPP problem defined on a data center that takes one of the topology in Figure 4, finding the TVMPP optimality is NP-hard.

Proof: This can be proved by a reduction from the Balanced Minimum K-cut Problem (BMKP) [18][19]. The BMKP problem is formally described as following: $G = (V, E)$ is an undirected, weighted graph with n vertices and n is a positive integer divisible by another positive integer k. A k-cut on G is defined as a subset of E that partition G into k components. The k-cut weight is defined as the sum of the weights on all the edges in the k-cut. The BMKP problem is to find a k-cut with minimum weight whose removal partitions G into k disjoint subsets of equal size $\frac{n}{k}$. BMKP is an extended problem from the Minimum Bisection Problem (MBP). Both BMKP and MBP are known to be NP-hard [18][19].

Now considering a data center network, regardless of which topology in Figure 4 being used, we can always create a network topology to satisfy the following requirements: there are n slots that are partitioned into k slot-clusters of equal size $\frac{n}{k}$ ($\frac{n}{k}$ is a positive integer). Every two slots have a connection with certain cost. While connections within the same cluster have equal cost c_i, connections across clusters have equal cost c_o with $c_o > c_i$.

Suppose there are n VMs with traffic matrix D. By assigning these n VMs to the n slots, we obtain a TVMPP problem. Meanwhile, if we define a graph with the n VMs as nodes and D as edge weights, we obtain a BMKP problem associated with the TVMPP problem. Any solution to the TVMPP also gives a solution to the associated BMKP problem. This is because the solution to the TVMPP partitions the n VMs into k groups, each group corresponding to a slot-cluster. The edges between all VM pairs that are assigned to two different slot-clusters can be considered as a k-cut, and the traffic between all such VM pairs are the k-cut weight. It can be shown that when the TVMPP is optimal, the associated BMKP is also optimal. Equivalently, when the TVMPP is optimal, if we swap any two VMs i, j that have been assigned to two slot-cluster r_1, r_2 respectively, the k-cut weight will increase. To prove this, we need to compute the amount of change occurring to the k-cut weight due to the swap of i, j. Clearly, this computation only needs involving those VM pairs of which one is i or j and the other is assigned to r_1 or r_2. Let s_1 denote the set of VMs assigned to r_1 (s_1 excludes the VM i and j). Analogously we define s_2. Now because the TVMPP objective value increases, we have

$$\sum_{\forall k \in s_1} [D_{jk}(c_i - c_o) + D_{ik}(c_o - c_i)] + \sum_{\forall k \in s_2} [D_{jk}(c_o - c_i) + D_{ik}(c_i - c_o)] > 0 \quad (3)$$

The amount of change for the k-cut weight is

$$[\sum_{\forall k \in s_1} (D_{ik} - D_{jk}) + \sum_{\forall k \in s_2} (D_{jk} - D_{ik})]$$

Due to (3) and $c_o > c_i$, it is straightforward to see that the above k-cut weight change is positive, i.e., before swapping VMs i, j, the BMKP problem achieves optimality. Thus, the k-cut is optimal when the TVMPP is optimal.□

The proof also reveals that the BMKP optimality is a necessary but insufficient condition for the associated TVMPP being optimal. This is because when BMKP is optimal, we can always swap two VMs both of which are assigned to the same slot-cluster. This does not affect the k-cut weight, but it can possibly decrease the TVMPP objective value.

IV. Algorithms

Previous analysis shows that the TVMPP problem is NP-hard and it belongs to the general QAP problem, for which no existing exact solutions can scale to the size of current data centers. Therefore, in this section we describe an approximation algorithm *Cluster-and-Cut* which leverages unique features of traffic patterns and network topologies in data centers. The proposed algorithm has two design principles, with the first one being the following well known result:

Proposition 1: [20] Suppose $0 \leq a_1 \leq a_2 \ldots \leq a_n$ and $0 \leq b_1 \leq b_2 \ldots \leq b_n$, the following inequalities hold for any permutation π on $[1, \ldots, n]$

$$\sum_{i=1}^{n} a_i b_{n-i+1} \leq \sum_{i=1}^{n} a_i b_{\pi(i)} \leq \sum_{i=1}^{n} a_i b_i$$

The TVMPP objective function is essentially to sum up all multiplications between every C_{ij} and its corresponding $D_{\pi(i)\pi(j)}$. According to Proposition 1, solving TVMPP is intuitively equivalent to finding a mapping of VMs to slots such that *VM pairs with heavy mutual traffic be assigned to slot pairs with low-cost connections*.

The second design principle is divide-and-conquer: we partition VMs into VM-clusters and partition slots into slot-clusters. Then we first map each VM-cluster to a slot-cluster. For each VM-cluster and its associated slot-cluster, we further map VMs to slots by solving another TVMPP problem, yet with a much smaller problem size. VM-clusters are obtained via classical min-cut graph algorithm which ensures that VM pairs with high mutual traffic rate are within the same VM-cluster. Such a feature is consistent with an early observation that traffic generated from a small group of VMs comprise a large fraction of the total traffic. Slot-clusters are obtained via standard clustering techniques which ensures slot pairs with low-cost connections belong to the same slot-cluster. Again, this is leveraging the fact that the network usually contains many groups of densely connected end hosts due to the star topology created by switches with many ports.

The pseudo-code for the algorithm is described in Algorithm 1. It has two major components:

1) SlotClustering: n slots are partitioned into k clusters by using the cost between slots as the partition criterion. There are two approaches in implementing this function. One is a manual procedure by the operators, who can leverage their *a prior* knowledge on network configurations. This approach may give better results but could be labor intensive. The other

Algorithm 1 Cluster-and-Cut

Require: D(Traffic matrix), C(Cost matrix), k(Number of clusters, a parameter used in clustering and min-cut components)

1: $n \leftarrow$ size of D {Find out VM count}
2: **SlotClustering(C,k)** {Partition slots into k clusters: $\{r_i\}$}
3: Sort $\{r_i\}$, in decreasing order of the cost of edges having one endpoint in r_i. Each edge only counts once
4: **VMMinKcut(**$D, \{|r_1|, \ldots, |r_k|\}$**)** {Partition n VMs into k clusters $\{s_i\}$, $|s_i| = |r_i|$}
5: Assign s_i to r_i, $\forall i = 1, \ldots, n$ {One-to-one mapping between slot-cluster and VM-cluster}
6: **for** $i = 1$ to k **do**
7: **if** $|s_i| > 1$ **then** {Multiple VMs in s_i}
8: Cluster-and-cut($D(s_i)$, $C(r_i)$, $|s_i|$){$D(s_i)$: traffic matrix for s_i. $C(r_i)$: cost matrix for r_i. $|s_i$: recursively call Cluster-and-Cut}
9: **end if**
10: **end for**

Algorithm 2 VMMinKcut

Require: G(Graph weight matrix), $\{b_1, \ldots, b_k\}$(size of each cluster)

1: $n \leftarrow$ size of G
2: Compute Gomory-Hu tree for G and obtain $n-1$ cuts $\{g_i\}$ {These $n-1$ cuts contains the minimum weight cuts for all server pairs}
3: Sort $\{g_i\}$ by increasing weight
4: **for** $i = 1$ to k **do**
5: Clear s_i
6: Find the minimum j such that removing $\{g_1, \ldots, g_j\}$ will partition G into two components: c_1 with size $|b_i|$ and c_2 with size $n - |b_i|$
7: $s_i \leftarrow c_1$
8: $G \leftarrow c_2$
9: $n \leftarrow n - |b_i|$
10: **end for**
11: Return $\{s_i\}$

approach is running classical clustering algorithms based on the cost matrix. Note that our cost definition is the number of switches on the path between two slots, so it satisfies the triangle inequality. Thus this becomes the Minimum k-clustering Problem [18] which is NP-hard. We solve this problem by the algorithm described in [21], with an approximation ratio 2. The output from SlotClustering is a set of slot-clusters, sorted in decreasing order of the total outgoing and incoming cost.

2) VMMinKcut: In this step, we need to partition n VMs into k VM-clusters with minimum inter-cluster traffic. More importantly, we must ensure that for any already formed slot-cluster, there is a corresponding VM-cluster with the equal size. The partition method used here is adapted from the minimum k-cut algorithm in [19]. This algorithm is originally applied to balanced Minimum k-cut problems in which the k clusters have equal size. The approximation ratio for the algorithm is $\frac{k-1}{k}n$. The pseudo-code of our adapted algorithm is described in Algorithm 2. A brief explanation is following: suppose we need to find a VM-cluster with certain size, we first find all the min-cut for every VM pair by applying the classical Gomory-Hu's algorithm [22]. It is shown in [22] that there are only $n-1$ distinct min-cut among the $\frac{n(n-1)}{2}$ total pairs of VMs. Then we find a subset of these $n-1$ min-cut such that their removal from G leads to a partition with the requested size. This process continues until all VM-clusters with requested size are formed. With similar proof as in [19], we can shown that this process terminates after finds k clusters with the same set of size as the previous k slot-clusters. Besides, the cut sorting and removal procedure ensures that smaller cost cuts have higher chance to be removed earlier. As a result, VM-clusters with low outgoing/incoming traffic more likely correspond to slot-clusters with low-cost outgoing/incoming connections. This complies with the aforementioned first design principle.

In Algorithm 1, after establishing a one-to-one assignment between slot-clusters and VM-clusters (Line 6), the rest of the code solves the VM-to-slot assignment problem within each cluster. Our strategy is the following: given the small cluster size $|s_i|$, we treat each single VM (or a single slot) as a cluster and run Cluster-and-Cut; if $|s_i|$ is still large, we can call Cluster-and-Cut to further divide s_i into smaller partitions and solve the problem recursively.

The computational complexity of this algorithm is determined by *SlotClustering* and *VMMinKcut*. SlotClustering has complexity $O(nk)$ [21]. VMMinKcut is reported to have complexity $O(n^4)$ [19]. Thus the total complexity is $O(n^4)$, which does not consider the recursive procedure.

V. Impact of Network Architectures and Traffic Patterns on Optimal VM Placement

We have discussed the advantage of using VM placement for improving network scalability. Through the problem formulation, we can notice that the traffic and cost matrices are the two determining factors for optimizing the VM placement. Consequently, we seek to answer a fundamental question: given that traffic patterns and network architectures in data centers have significant differences, how the performance gains due to optimal VM placement are affected. Answering this question not only allows us to better understand the value and limit of our approach, it also benefits the design of future data center network architecture in general.

Since computing the TVMPP optimality with general D and C is intractable, to gain insight, we focus on two special traffic models : 1) global traffic model in which each VM communicates with every other at a constant rate; 2) partitioned traffic model in which VMs form isolated partitions, and only VMs within the same partition communicate with each other. The global model is one of the very few cases that we can obtain the TVMPP optimality in polynomial time. Traffic patterns in reality can be roughly considered to be generated from a mixture of these two special models.

We also need to derive the cost matrix C under various network architectures. For this purpose, we develop analytical expressions of C for the four architectures described in Section II. For the Tree topology, the cost between two VMs is a function of the fan-out of the access switches (p_0) as well as the fan-out of the aggregation ones (p_1):

$$C_{ij}^{Tree} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } \lfloor \frac{i}{p_0} \rfloor = \lfloor \frac{j}{p_0} \rfloor \\ 3 & \text{if } \lfloor \frac{i}{p_0} \rfloor \neq \lfloor \frac{j}{p_0} \rfloor \wedge \lfloor \frac{i}{p_0 p_1} \rfloor = \lfloor \frac{j}{p_0 p_1} \rfloor \\ 5 & \text{if } \lfloor \frac{i}{p_0 p_1} \rfloor \neq \lfloor \frac{j}{p_0 p_1} \rfloor \end{cases}$$

In the VL2 architecture, the cost is a function only of the fan-out of the access switches (p_0), given that traffic that leaves the access switches always goes through the core switches (due to the valiant load balancing):

$$C_{ij}^{VL2} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } \lfloor \frac{i}{p_0} \rfloor = \lfloor \frac{j}{p_0} \rfloor \\ 5 & \text{if } \lfloor \frac{i}{p_0} \rfloor \neq \lfloor \frac{j}{p_0} \rfloor \end{cases}$$

In the PortLand Fat-tree architecture, the cost is a function of k, the total number of ports on each switch:

$$C_{ij}^{Fat-tree} = \begin{cases} 0 & \text{if } i = j \\ 1 & \text{if } \lfloor \frac{2i}{k} \rfloor = \lfloor \frac{2j}{k} \rfloor \\ 3 & \text{if } \lfloor \frac{2i}{k} \rfloor \neq \lfloor \frac{2j}{k} \rfloor \wedge \lfloor \frac{4i}{k^2} \rfloor = \lfloor \frac{4j}{k^2} \rfloor \\ 5 & \text{if } \lfloor \frac{4i}{k^2} \rfloor \neq \lfloor \frac{4j}{k^2} \rfloor \end{cases}$$

In the BCube architecture, the cost is a function of the Hamming distance of server addresses:

$$C_{ij}^{BCube} = \begin{cases} 0 & \text{if } i = j \\ 2\, hamming(addr(i), addr(j)) - 1 & \text{if } i \neq j \end{cases}$$

A. Global Traffic Model

Under the global traffic model, each VM sends traffic to every other VM at equal and constant rate, whereas this sending rate can differ among VMs. Accordingly, the traffic matrix D consists of constant row vectors. For any permutation matrix X, $DX^T = D$ holds. This simplifies the TVMPP problem in the form of (2) to the following

$$\min_{X \in \Pi} S = tr(DC^T X) \tag{4}$$

which is the classical Linear Sum Assignment Problem (LSAP) [23]. The complexity for LSAP is $O(n^3)$ and there exist a number of efficient algorithms. Here we apply the Hungarian algorithm to find its optimal solution.

We then consider a random placement in which any VM has equal probability to be assigned to any slot. Such a random placement reflects assignment of VMs that does not take advantage of the TVMPP optimization. Let S_{rand} denote the expected objective value achieved by a random placement. It can be easily computed from (4) as

$$S_{rand} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} (i,j) \text{ entry in } DC^T$$

We now compare S_{opt} with S_{rand} under various settings. Figure 5 shows their comparison under the four network architectures, in a network of 1024 VMs. In the figure, each bar is an overlapping of S_{opt} and S_{rand}. Because $S_{rand} \geq S_{opt}$ always holds, the gap appearing on the top of each bar represents the improvement space for a random placement. In Figure 5, we let the four traffic matrices follow the global traffic model, with the total outgoing traffic from each VM following a normal distribution with a mean of 1 and a variance of 0, 0.25, 0.5 and 0.75.

When all the entries of the traffic matrix are equal, corresponding to variance 0, the random placement achieves the optimal value, given that the gap between S_{rand} and S_{opt} is zero. As the traffic variance increases, we notice that the gap becomes larger. This indicates that under a global traffic model, a random VM placement has improvement space, and this space increases with the increase of traffic variance. In other words, if a data center is devoted to just one application with homogeneous traffic pattern among VMs, such as a map-reduce type of workload, then traffic-aware placement of the VMs provides little improvements. In contrast, network scalability improvements are greater for data centers devoted to workloads with very heterogeneous traffic among VMs, such as ones produced by multi-tiered web applications.

Moreover, the figure shows that the improvement space varies among the four architectures: While the BCube has the largest improvement space, the VL2 comes with the smallest. The BCube sees the most benefit mainly because for a system of 1024 VMs, the required BCube network has four levels of intermediate switches, which results in larger path cost. Thus, optimizing VM placement based on network traffic yields a higher gain. In contrast, almost every element of C in the VL2 is equal (due to the valiant load balancing) and as such any efforts in optimizing VM placement is almost fruitless. Nevertheless, such an argument does not necessarily mean the BCube architecture is more scalable than the VL2. It depends on many other factors, with most notably the network oversubscription. Our results only indicate that a BCube architecture can greatly benefit in terms of its scalability with TVMPP, while the VL2 sees the smallest benefit.

B. Partitioned Traffic Model

Under the partitioned traffic model, each VM belongs to a group of VMs and it sends traffic only to other VMs in the same group, with the pairwise traffic rate following a normal distribution. Computing S_{opt} in this case requires an exact solution, which becomes prohibitively expensive for problem sizes larger than 15 VMs/slots. As such we report here results by replacing $Sopt$ by the classical Gilmore-Lawler Bound (GLB) [24]. The GLB is a lower bound for the optimal objective value of a QAP problem and it can be computed reasonablely fast. When the objective value for a solution is close to the GLB, it suggests that the space for improving that solution is limited. On the other hand, if the objective value of a solution is far higher than the GLB, it does not necessarily mean the solution is much worse than the optimal one. Rather, the only conclusion we can draw is that the performance improvement potential for that solution is high.

Figure 6 compares for the four architectures the GLB against the objective value of a random placement in a system of 1024 VMs, when there are 16 groups of size 64 VMs each.

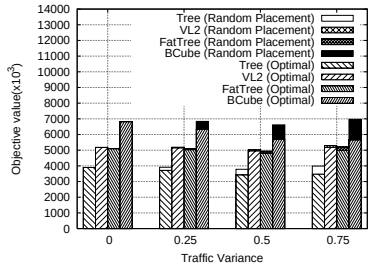


Fig. 5. Optimal objective value *vs* objective value achieved by random placement (global traffic model, different traffic variance)

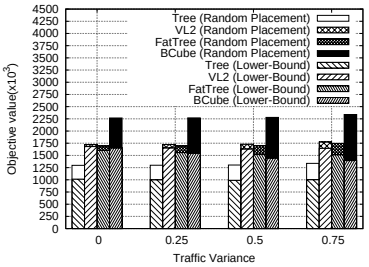


Fig. 6. GLB *vs* objective value achieved by random placement (partitioned traffic model, 16 partitions of 64 VMs each)

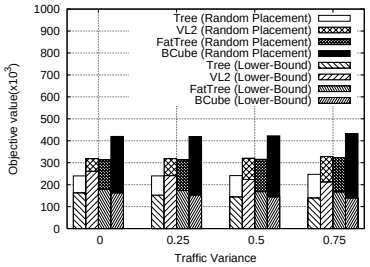


Fig. 7. GLB *vs* objective value achieved by random placemente (partitioned traffic model, 10 partitions with 2^i VMs in each, $i = 1, .., 10$)

Figure 6 shows that even when the pairwise traffic rates are equal within each partition (namely, zero variance), the random placement comes with a performance gap relative to the GLB. As in the case of the global traffic model, this gap increases when the traffic variance within each partition increases. In addition, the figure shows that the gap is larger under the BCube architecture compared to the other ones, for the same reasons as in the case of global traffic model.

Figure 7 shows similar results for a system of different partition size. More specifically, the results are for a system of 10 groups of 2^i VMs each ($i = 1, .., 10$). We observe the same trends as in the previous two cases, with the performance improvement potential being even more prominent. This can be partially attributed to the existence of small partition sizes. Indeed, Figure 8 compares different partition sizes, with equal pairwise traffic rates within each partition. It is clear that the improvement potential is proportionally higher for smaller partition sizes. When the partition size reaches the total system size the improvement potential becomes zero, given the equal pairwise traffic rates within the partition.

C. Summary

To summarize, the above results provide the following insights:

- The potential benefit of optimizing TVMPP is greater with increased traffic variance within one partition, i.e. one composite application. This is attributed to the fact that VMs with high pairwise traffic volume are placed close to each other.
- The potential benefit of optimizing TVMPP is greater with increased number of traffic partitions, i.e. number of isolated composite applications, or decreased partition size, i.e. applications running on less VMs. This is attributed to the fact that VMs belonging to the same composite application are placed close to each other.
- The potential benefit of optimizing TVMPP depends on the network architecture. The benefit is greater for a multi-layer architecture, such as BCube; the benefit is minimal for an architecture that employs network load balancing techniques, such as in VL2.

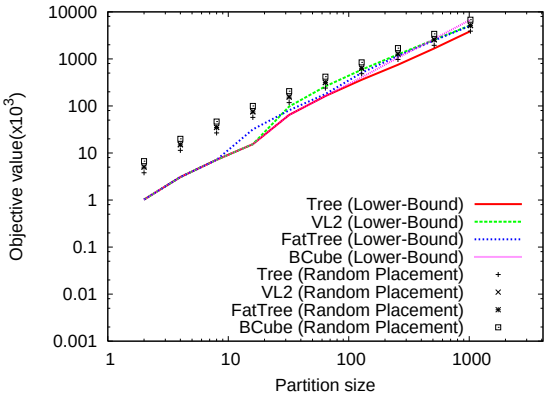


Fig. 8. GLB *vs* objective value of random placement (with different partition size)

VI. Evaluation of Algorithm Cluster-and-Cut

A. Experiment Settings

We compare the proposed Cluster-and-Cut algorithm and two benchmark algorithms in a number of trace-driven settings. There are many heuristics for general QAP problems. Among them, we select two representative ones: Local Optimal Pairwise Interchange (LOPI) [25] and Simulated Annealing (SA) [26]. The detailed descriptions are referred to their original papers.

We use the equations in Section V to generate the cost matrices C. For D, we consider a hybrid traffic model which combines real traces with the classical *Gravity model* [27]. The trace collected from production data centers (mentioned in Section II) provides the aggregate incoming and outgoing traffic rates at VMs. To decompose it into rates between VMs, we assume the Gravity model holds in data centers, i.e., the rate from VM i to j is decided by $D_{ij} = \frac{D_i^{out} D_j^{in}}{\sum_k D_k^{in}}$, where D_i^{out} is the total outgoing rate at VM i, D_j^{in} is the total incoming rate at VM j. Obviously, the Gravity model tends to produce global traffic pattern. For comparison purposes, we also compute the GLB for each settings.

B. Experiment Results

We compare the above three algorithms by using the hybrid traffic model. The number of slots (and VMs) is 1024. The results are summarized in Table I. It is noticeable that the

Topology	Algorithms	Gilmore-Lawler bound	Performance	
			Best value	CPU min
Tree	LOPI	4.63e+10	8.22e+10	22
	SA		8.35e+10	27
	Cluster-and-Cut		8.13e+10	11
VL2	LOPI	7.03e+10	1.09e+11	25
	SA		1.12e+11	31
	Cluster-and-Cut		1.05e+11	12
Fat-tree	LOPI	6.43e+10	1.07e+11	26
	SA		1.12e+11	32
	Cluster-and-Cut		0.97e+11	13
BCube	LOPI	5.55e+10	1.43e+11	29
	SA		1.41e+11	35
	Cluster-and-Cut		1.21e+11	14

TABLE I
ALGORITHM PERFORMANCE WITH HYBRID TRAFFIC MODEL

objective function value given by the Cluster-and-Cut is about 10% smaller than the two benchmarks, with CPU time being halved. This experiment demonstrates the efficacy of the proposed Cluster-and-Cut algorithm.

VII. DISCUSSION

Combining VM migration with dynamic routing protocols: We have assumed static layer 2 and 3 routing protocols. From the perspective of our problem formulation, a dynamic routing means the cost matrix may be changing. This could happen either due to topology changes or due to the routing protocols being traffic aware. Such a change may potentially improve the TVMPP objective value without requiring shuffling VMs. However, such an improvement has a limit. E.g., two VMs with high mutual traffic may reside in two physically remote LANs, in which case the dynamic routing can not benefit much.

VM placement by joint network and server resource optimization: We have considered the VM placement problem only with respect to network resource optimization. Previous approaches have considered the VM placement problem with respect to server resource optimization, such as power consumption or CPU utilization. The formulation of a joint optimization of network and server resources is still an open problem, with many practical applications. For example, minimizing the total energy consumption in a data center requires the formulation of a joint optimization problem.

VIII. CONCLUSION

This paper presents an approach of manipulating VM placement to address the scalability concern in modern data center networks. A careful VM placement can localize large chunks of traffic and thus reduce load at high-level switches. We formulate the Traffic-aware Virtual Machine Placement Problem (TVMPP), prove its NP-hardness and propose a two-tier approximation algorithm to efficiently solve it. Another major result is an analysis on how traffic patterns and the network topology in data centers affect the potential network scalability benefit by optimally solving the TVMPP. This analysis considers four representative data center network topologies. The analysis shows the value and limit of using VM placement to improve network scalability under different network topologies and traffic patterns.

REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. ACM, 2008, pp. 63–74.

[2] N. F. N. H. P. M. S. R. V. S. Radhika Niranjan Mysore, Andreas Pamboris and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009.

[3] 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 *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009.

[4] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. ACM, 2008, pp. 75–86.

[5] ——, "Bcube: A high performance, server-centric network architecture for modular data centers," in *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. ACM, 2009.

[6] 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, no. 1, pp. 68–73, 2009.

[7] J. P. Srikanth kandula and P. Bahl, "Flyways to de-congest data center networks," in *ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.

[8] VMware Capacity Planner, "http://www.vmware.com/products/capacity-planner/."

[9] IBM WebSphere CloudBurst, "http://www-01.ibm.com/software/webservers/cloudburst/."

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

[11] Lanamark Suite, "http://www.lanamark.com/."

[12] "Cisco data center infrastructure 2.5."

[13] W. J. Dally and B. Towles., "Principles and practices of interconnection networks," in *Morgan Kaufmann Publishers*, 2004.

[14] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," in *IEEE Transactions on Computers*, 1985.

[15] L. Bhuyan and D. Agrawal., "Generalized hypercube and hyperbus structures for a computer network," in *IEEE Transactions on Computers*, 1984.

[16] P. O. B.-N. P. H. Eliane Maria Loilola, Nair Maria Maia de Abreu and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, pp. 657–690, 2007.

[17] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the Association of Computing Machinery*, vol. 23, pp. 555–565, 1976.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H.Freeman, 1979.

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

[20] G.G.Hardy, J.E.Littlewood, and G.Polya, *Inequalities*. London and New York: Cambridge University Press, 1952.

[21] T. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[22] R. E. Gomory and T. C. Hu, "Multi-terminal network flows," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 551–570, 1961.

[23] R. E. Burkard and E. la, "Linear assignment problems and extensions," 1998.

[24] P. Gilmore, "Optimal and sub-optimal algorithms for the quadratic assignment problem," *J. SIAM*, pp. 305–313, 1962.

[25] G. C. Armour and E. S. Buffa, "A heuristic algorithm and simulation approach to relative location of facilities," *Management Science*, vol. 9, no. 2, pp. 294–309, 1963.

[26] E. Burkard and F. Rendl, "A thermodynamically motivated simulation procedure for combinatorial optimization problems," *European Journal of Operation Research*, vol. 17, pp. 169–174, 1984.

[27] Y. Zhang, M. Roughan, N. Duffield, and A. Greeberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," in *ACM SIGMETRICS*, 2003, pp. 206–217.