2016 IEEE 2nd International Conference on Big Data Security on Cloud, IEEE International Conference on High Performance and Smart Computing, IEEE International Conference on Intelligent Data and Security

Traffic Aware Virtual Machine Packing in Cloud Data Centers

Jun Liu, Jinhua Guo, and Di Ma
Department of Computer and Information Science
University of Michigan-Dearborn
Dearborn, Michigan 48128, USA
Email:{juliu,jinhua,dmadma}@umich.edu

***Abstract*—This paper investigates the virtual machine packing problem in cloud data centers. A cloud tenant expresses computation requirement for each virtual machine (VM) and bandwidth requirement for each pair of VMs. A cloud provider places the VMs and routes the traffic among the VMs in a way that minimizes the total number of servers used while providing both computation and bandwidth guarantees. We formulate a special Multi-Capacity Bin Packing problem that consolidates VMs into the smallest number of servers. A server is represented by a bin with multiple capacities corresponding to the computation resource (CPU, memory, and storage) and communication bandwidth resource available. Unlike bin packing, in the traffic-aware VM packing, the cumulative bandwidth requirement of all VMs hosted in a server can be smaller than the sum of bandwidth demand of all VMs due to the co-location. Our experimental results show the proposed solutions significantly reduce the number of servers needed as well as the total inter-server traffic, and will have great potential for reducing energy consumption in data centers.**

***Keywords*—Network Virtualization, Multi-capacity Bin Packing, Energy Efficiency, Virtual Machine Packing, Flow Routing**

I. Introduction

Virtualization in data centers is an efficient way to increase resource utilization and reduce energy consumption. Server virtualization allows multiple virtual machines (VMs) to be co-located in a single physical server. Network virtualization aims at creating multiple virtual networks on top of a shared physical network substrate allowing each virtual network to be implemented and managed independently. With such virtualization, the resources can be scheduled with fine-granularity, which improves resource utilization significantly.

However, mapping multiple virtual networks into a data center network is very challenging. The data center must provide all resources that each virtual network requires, including computation resource (CPU, memory, and storage) for each VM and communication bandwidth for each pair of VMs. Currently, most cloud data centers offer only guarantees on the computation requirement. However, networks are shared among tenants in a best-effort manner. Consequently, tenants often experience variable and unpredictable network performance [1].

With the explosive growth of data center traffic, more and more communication-intensive applications require inter-VM connections with very high bandwidth. The number of VMs can be hosted by a server or physical machine (PM) is often constrained by the bandwidth capacity instead of the computation capacity. In this paper, term "server" and "physical machine (PM)" are interchangeable. Although the communication bandwidth among VMs placed in the same server is almost unlimited, the total ingress and egress traffic of all VMs must not exceed the bandwidth capacity of the server adapter.

In this paper, we investigate the traffic-aware virtual machine packing problem in cloud data centers. The objective is to minimize the number of physical machines (PMs) required to host all the VMs as well as minimize the inter-PM traffic. Most existing virtual network mapping solutions ignore the virtual machine (VM) packing problem. They simply assume that each VM has the same size, and each PM supports only one VM [2] or a fixed number of VMs [3], [4].

We partition VMs into VM-clusters. Each VM-cluster will be hosted in a single server. This can be generalized to solving a *Multi-Capacity Bin Packing problem* [5]. A server is represented by a bin with multiple capacities corresponding to the available computation resource of the server and bandwidth resource of the server adapter. Unlike bin packing, in the traffic-aware VM packing, the cumulative bandwidth requirement of all VMs hosted in a server can be smaller than the sum of individual VM bandwidth demand because the communications between co-located VMs are through the internal channels and do not require any bandwidth from the server adapter. The efficient VM packing must identify heavy communication patterns and then place VMs with heavy traffic loads in the same server or if not possible in close distance in data centers. Since this problem is NP hard, we propose an approximation algorithm based on the weighted graph cutting. Our solution minimizes both the number of active servers and inter-server traffic, thus greatly reduce the energy consumption in data centers.

The rest of this paper is organized as follows. Section II describes related work. The system model is formally defined in Section III. Section IV presents our solutions and the details of the algorithms. The performance evaluation results are discussed in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

A. Virtual Machine Packing/Consolidating

To increase resource utilization and energy efficiency, VMs are often consolidated into a smaller number of physical servers in a way that minimizes the number of servers used.

978-1-5090-2403-2/16 $31.00 © 2016 IEEE
DOI 10.1109/BigDataSecurity-HPSC-IDS.2016.37

Traditional solutions consider only capacity limit of computation resources including CPU, memory, and storage [6]. However, with the dramatic growth of data center traffic, network bandwidth capacity limit must be taken into consideration. The aggregate traffic rate for VMs placed on the same server must not exceed the adapter bandwidth capacity. Wang et al. [7] formulate the VM consolidation as a Stochastic Bin Packing problem which models the bandwidth demands of VMs as probabilistic distributions and propose an online packing solution. However, they do not consider that the cumulative bandwidth requirement of VMs hosted in a server can be smaller than the sum of individual VM bandwidth requirement due to the co-location.

In contrast, our VM packing solution takes into consideration both computation and bandwidth capacity limits as well as the co-location bandwidth reduction.

B. Traffic Aware Virtual Machine Placement

Some recent research works have studied VM placement for minimizing total communication cost between VM pairs in data centers. Meng et al. [3] and VM Planner [4] investigate traffic-aware virtual machine placement in data center networks. The idea is to place VM pairs with heavy traffic among them on servers with small communication cost.

However, they assume that each virtual machine has the same size, and each server supports only one VM [2] or a fixed number of VMs [3], [4].

C. Virtual Network Models

Network virtualization with bandwidth guarantees in data centers is very complex and challenging. There are two popular virtual network abstractions: hose model and pipe model.

In the hose model abstraction, such as Oktopus [1] and ElasticSwitch [8], all VMs are connected to a central (virtual) switch by a dedicated link (hose) having a minimum bandwidth guarantee. This model mimics dedicated clusters with compute nodes connected through Ethernet switches. However, it does not accurately express the requirements for applications with complex traffic interactions.

In the pipe model abstraction, such as SecondNet [2] and CloudNaaS [9], it specifies bandwidth guarantees between pairs of VMs as virtual pipes. When the traffic is predicable and relative stable, this model can precisely capture the application traffic needs. In this paper, we are using the pipe model to describe the traffic needs of tenants.

Further, Qiu et al. [10] propose a genetic-based optimization algorithm for chip multiprocessor (CMP) equipped with phase-change memory (PCM) memory, which improves energy efficiency in cloud computing.

III. The System Model

In this study, a tenant request is abstracted to a virtual network using pipe model. A virtual network is modeled as a graph G_v with a set of VMs T as vertices and a traffic matrix M as edges. A tenant can specify a diverse set of computation and bandwidth requirements. Thus, the modeled graph is a weighted graph. The vertex weight represents the VM computation demand r_i, and the edge weight represents the traffic flow size $m_{i,j}$ between VMs i and j.

The VM placement is restricted by two constraints: PM computation capacity C_s and link bandwidth capacity C_l. We define the ***load*** of PM p as $\sum_i \Delta_{i,p} r_i$, which should be no more than its computation capacity C_s, ($\sum_i \Delta_{i,p} r_i \leq C_s$).

TABLE I. Notation

T	Set of VMs
P	Set of PMs
M	Traffic matrix of a tenant virtual network
$m_{i,j}$	Bandwidth demand by the flow between VM $i \in T$ and VM $j \in T$
r_i	computation demand by VM i
$\Delta_{i,p}$	A binary decision for assigning VM i to PM p
y_p	A binary decision for PM p being active.
C_s	computation capacity of a PM
C_l	bandwidth capacity of a link
W	Set of VM-clusters
D	inter VM-cluster traffic matrix
$d_{i,j}$	Bandwidth required by the flow between VM-cluster $i \in W$ and VM-cluster $j \in W$

We partition graph G_v into subgraphs. Each subgraph is considered as a VM-cluster, which can be hosted in a single PM. A VM-cluster is a subset T' of VMs in T with high intra-group traffics, ($T' \subset T$). $T - T'$ is the subset of VMs outside the cluster in T. We let ***inter-cluster traffic*** of T' to be traffics between T' and $T - T'$, which is $\sum_{i \in T', j \in T - T'} m_{i,j}$. Therefore, we can minimize the number of PMs by minimizing the number of VM-clusters under the constraints C_s and C_l, see equation (1).

$$\text{Minimize} \quad \sum_{p \in P} y_p \tag{1a}$$

$$\text{Subject to} \quad \sum_{i \in T} \Delta_{i,p} r_i \leq C_s * y_p, \tag{1b}$$

$$\sum_{p \in P} \Delta_{i,p} = 1, \tag{1c}$$

$$\sum_{i,j \in T} m_{i,j} * (\Delta_{i,p}(1 - \Delta_{j,p}) + \Delta_{j,p}(1 - \Delta_{i,p})) \leq C_l. \tag{1d}$$

Constraint (1b) models the computation capacity constraint of a PM where VMs are placed. Constraint (1c) means any VM must be placed into one and only one PM. Constraint(1d) captures bandwidth constraint of a PM network adapter.

IV. Traffic-Aware Virtual Machine Packing

VM packing falls into the class of Multi-Capacity Bin Packing Problem [5], which is a known NP-complete problem. Unlike bin packing, in the traffic-aware VM packing, the cumulative bandwidth requirement by a cluster of VMs hosted in a server can be smaller than the sum of individual VM bandwidth requirement, as shown in Constraint (1d). This is because the communications between co-located VMs are

through the internal channels and do not require any communication bandwidth from the server adapter.

For a communication-intensive virtual network, it is the link bandwidth that is the bottleneck to consolidate VMs into PMs. Therefore, to minimize the number of PMs, we must minimize the inter VM-cluster traffic.

We propose Algorithm VM-Packing (see Algorithm 1) to identify VM-clusters by using traffic matrix M while minimizing inter VM-cluster traffics. Algorithm VM-Packing cuts G_v into VM-clusters, which can be hosted by PMs under the constraints of C_s and C_l. When VM-clusters are mapped into PMs, all inter VM-cluster traffics are carried out by data center network while intra VM-cluster traffics are through the internal channels and thus can be ignored.

There are two steps in Algorithm VM-Packing. The first step (Line 3-21) is to identify traffic patterns by traffic matrix M, and the second step (Line 22-37) is to pack the traffic patterns into VM-clusters which can be hold by PMs under the constraints of C_s and C_l. In the first step, we have two operations: merge and partition. In the merge operation (Line 4-8), we sort the flows $m_{i,j}$ by their sizes in non-increasing order. We get the largest flow $m_{u,v} = \max_{i,j \in T} m_{i,j}$ and two associated VMs u and v. If $r_u + r_v \leq C_s$ and $\sum_{i \in T} m_{u,i} + \sum_{i \in T} m_{v,i} - 2m_{u,v} \leq C_l$, we merge VMs u and v into a new super VM, and then recompute the traffic matrix M. This newly created super VM is a traffic pattern. We continue this operation until no VM pair can be merged. Each time we recompute traffic matrix M, and update the virtual network graph G_v.

After merge operation, we do partition operation (Line 9-20). We generate a Maximum Spanning Tree (MST) for the merged graph G_v. M^{MST} is the traffic matrix of the MST. We sort the flows $m_{i,j}^{MST}$ by their sizes in non-decreasing order. We remove the smallest edge $e_{u,v} = \min_{i,j \in T} m_{i,j}^{MST}$, and partition the MST into two subtrees MST_1 and MST_2 where T_1 and T_2 are subsets of VMs in MST_1 and MST_2 respectively. For each subtree MST_k and T_k, if $\sum_{i \in T_k} r_i \leq C_s$ and inter-pattern traffic of T_k $\sum_{i \in T_k, j \in T - T_k} m_{i,j} \leq C_l$, we merge VM set T_k into a new super VM; otherwise, partition MST_k by removing $e_{u,v} = \min_{i \in T_k, j \in T_k} m_{i,j}^{MST_k}$ until CPU utilization of each partitioned pattern is no more than C_s and its inter-pattern traffics is no more than C_l. Then we recompute traffic matrix M of the updated G_v. We repeat merge and partition operations until no new super VM is created in the first step. After the first step, we move to the second step to pack the VM patterns into VM-clusters using the First-Fit (FF) bin packing algorithm .

For example, we have a virtual network graph with 17 VMs, as shown in see Fig. 1. Let's assume the computation capacity $C_s = 12$ and the link bandwidth capacity $C_l = 24$. We let $r_i = 1$ for $1 \leq i \leq 12$, $r_i = 3$ for $13 \leq i \leq 17$, and $\sum_{j \in T} m_{i,j} \leq C_l$ for $1 \leq i \leq 17$. In the first round merge operation, VMs 13 and 14 are merged, and VMs 15 and 16 are merged(see Fig. 2). Then we generate a MST tree for the graph, (see Fig. 3), run the partition operation to get three new super VMs, VM $1, 2, 3$, VM $4, 5, 6$, and VM $7, 8, 9$, see (Fig. 4). We continue the next round operations, and get two new super VMs, VM $1, 2, 3, 7, 8, 9, 11$ and VM $4, 5, 6, 10$ by the merge operation, (see Fig. 5). Finally, we get a new super VM

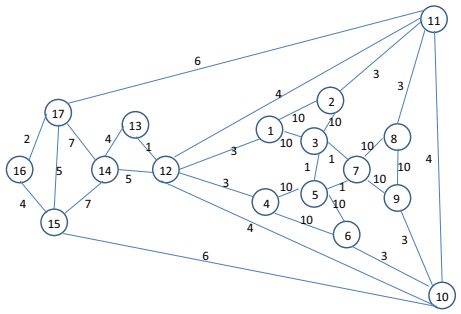

Fig. 1. Graph of a tenant virtual network with 17 nodes while $C_s = 12$ and $C_l = 24$

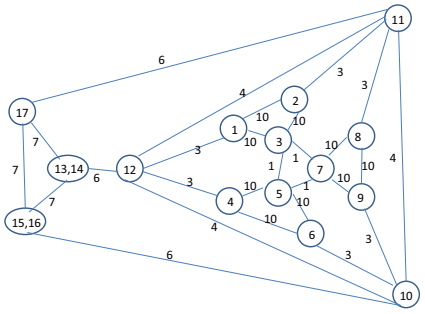

Fig. 2. Merge VMs 13 and 14, VMs 15 and 16 respectively

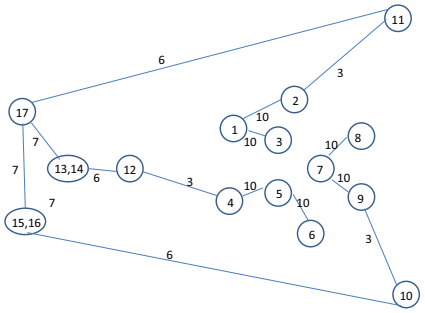

Fig. 3. Maxium Spanning Tree of the graph

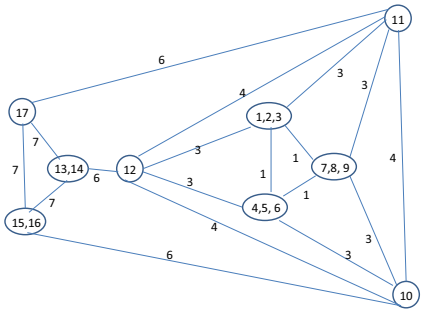

Fig. 4. Partition VMs $1, 2, 3$, VMs $4, 5, 6$, VMs $7, 8, 9$, and then merge them into three super VMs

Algorithm 1 VM-Packing

Input: Graph G_v with a set of VMs T as vertices and traffic matrix M as edges
Output: VM-cluster set $W = \{W_1, W_2, ..., W_n\}$, and inter VM-clusters traffic matrix D

Set $W_i = \phi$,$i = 1, ..., n$
Set $D = 0$
repeat
  sort $m_{i,j}$ with $i, j \in T$ in non-increasing order //Start operation merge
  while existing VM pairs can be merged under the constraints C_l and C_s **do**
    merge VMs u and v with $m_{u,v} = \max_{i,j \in T} m_{i,j}$ into a super VM
    sort $m_{i,j}$ with $i, j \in T$ in non-increasing order
  end while// End operation merge
  generate Maximum Spanning Tree MST for the graph G_v//Start operation partition
  enqueue MST to queue Q
  while Q is not empty **do**
    dequeue the first element MST' from Q where T' is the VM set of MST'
    if $\sum_{j \in T'} r_i \leq C_s$ and inter-pattern traffic requirement of T' $\sum_{i \in T', j \in T - T'} m_{i,j} \leq C_l$ **then**
      merge VMs in T' into a super VM
    else
      partition MST' by removing the edge with minimum weight in MST'
      put all partitioned trees into Q
    end if
  end while
  recompute the traffic matrix M by putting super VMs back into G_v and recovering the removed edges//End operation partition
until no super VM is created
{Packing VMs by the First-Fit Algorithm in the following statements}
sort VMs in T by their CPU utilization // T is updated by merge and partition operations
for each VM i with $i \in T$ **do**
  for each Bin j **do**
    if Bin j can hold VM i under the constraints C_s and C_l **then**
      $W_j \Leftarrow W_j \cup$ VM i // W_j is subset of VMs in Bin j
    end if
  end for
  if no Bin can hold VM i **then**
    initialize an empty bin n, $W_n \Leftarrow$ VM i
  end if
end for
compute traffic matrix D among bins

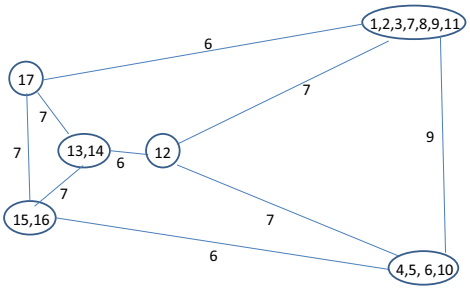

Fig. 5. The second round merging

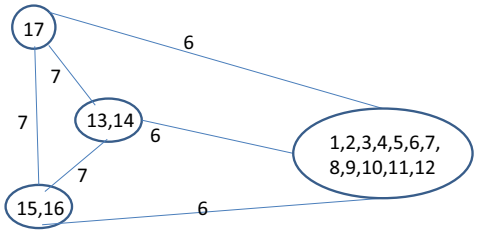

Fig. 6. The second round partitioning

$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12$ by the partition operation, (see Fig. 6).

Theorem 1: The complexity of Algorithm VM-Packing is $O(N^3 \log^2 N)$ where $N = |T|$.

Proof: In the first step of Algorithm VM-Packing, two operations merge and partition are executed in a loop. Operation merge is also a loop, and edges are sorted in non-increasing order. The time cost is $O(N^2 \log N)$. Then the largest edge is merged, and traffic matrix M is updated. The time cost is N. The merge loop is executed at most N times, so the time cost for operation merge is $N^3 \log N$.

For partition operation, the time cost of generating MST is $O(N^2 \log N)$. The loop of removing an edge in MST is N times. For each removing, checking subtree is performed no more than N^2 times, so the time cost of partition operation is N^3.

The loop of the first step of Algorithm VM-Packing is executed in $\log N$ times. This can be proved in the follow way. We assume the generated MST of graph G_v is composed of such edges: $e_{r,s} \geq e_{s,t} \geq e_{t,u} \geq e_{u,v} \geq e_{v,w} \leq e_{w,x} \leq e_{x,y}$ where VM r is the root of MST and edge $e_{v,w}$ is the smallest. In partition operation, we partition MST tree by removing the smallest edge $e_{v,w}$, then MST is divided into two parts MST_i and MST_j where MST_j is composed of $[e_{w,x}, e_{x,y}]$. We assume VMs in MST_j satisfy constraints of C_s and C_l. The number of edges in MST_j is either 0 or at least 2 because if one edge in MST_j, this edge must be merged in the merge operation. After this partition, since the number of edges is at least two, the newly created super VM w^* must contain over three VMs. The MST is changed into $e_{r,s} \geq e_{s,t} \geq e_{t,u} \geq e_{u,v} \geq e_{v,w^*}$.

In the next round merge operation, if no VM is merged, the MST dose not change, thus no new super VM can be created and the loop stops. In order to continue the loop, at least two new super VMs v^* and w^* must be created by merge operation to update MST. We assume the updated MST is $e_{r,s}, e_{s,t}, e_{t,u}, e_{u,v^*}, e_{v^*,w^*}$. In this updated MST, if we have $e_{r,s} \geq e_{s,t} \geq e_{t,u} \leq e_{u,v^*} \leq e_{v^*,w^*}$, the partition operation can partition the updated MST and create new super VM. For newly created super VMs v^* and w^*, each of them are merged by VMs, at least one of which must be the new super VM created in previous round.

For the $n-th$ round of merge and partition loop, in order to continue the loop, at least two new super VMs must be created by merging at least four VMs and two of them must be super VMs created in the previous round, so the number of VMs is reduced at least 2^n times in the $n-th$ round of the loop. Therefore, we derive the loop is executed in no more than $\log N$ times.

The time cost of Algorithm FF is no more than N^3, so the complexity of Algorithm VM-Packing is $O(N^3 \log^2 N)$.

■

V. Performance Evaluation

We evaluate the performance of Algorithm VM-Packing by simulations. The traffic matrix is generated using log-normal distribution with density function shown in Equation (2). In all tests of log-normal distribution, the mean μ is set to 0 while the standard deviation σ is set to 1 or 1.5. Increasing σ, we create a more scattered and higher total traffic between VMs.

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \tag{2}$$

We test three cases of edge weight distributions: two log-normal distribution where σ is set to 1 and 1.5 and one standard normal distribution where edge weight varies between $[1,2,3,4,5]$ randomly.

Algorithm VMP [11] is the only other solution that considers the VM packing together with the VM placement. So, we compare Algorithm VM-Packing with Algorithm VMP. Both algorithms identify patterns based on the traffic matrix. However, Algorithm VMP does not have the merge step as Algorithm VM-Packing does.

We let the PM computation capacity C_s to be 200, and link bandwidth capacity C_l to be 100. In the graph of a virtual network, the degree of a node varies from 0 to $|T|-1$. For each VM $i \in T$, we assume $r_i \leq C_s$ and $\sum_{j \in T} m_{i,j} \leq C_l$; otherwise a tenant's request for the virtual network may be denied.

Both algorithms output a set of VM-clusters W and an inter VM-cluster traffic matrix D. Since each VM-cluster is hosted by one PM, the number of PMs needed equals to $|W|$ and the inter-PM traffic matrix is the same as that of VM-clusters. We let W' and W be the set of PMs which host VM-clusters and D' and D be the inter-PM traffic matrix resulted from Algorithm VMP and Algorithm VM-packing, respectively. We define the PM ratio η_s to be $\frac{|W'|}{|W|}$ and the inter-PM traffic ratio η_t to be $\frac{\sum_{i,j \in W'} d'_{i,j}}{\sum_{i,j \in W} d_{i,j}}$.

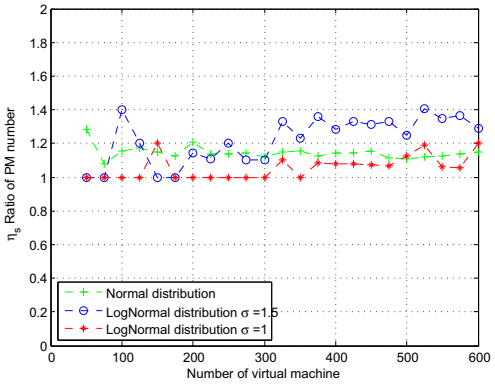

Fig. 7. The PM ratio when the number of VMs ranges from 50 to 600 stepped by 25, and the mean degree of VMs is 8.

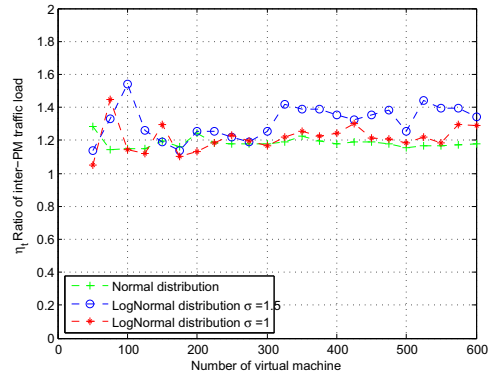

Fig. 8. The inter-PM traffic ratio when the number of VMs ranges from 50 to 600 stepped by 25, and the mean degree of VMs is 8.

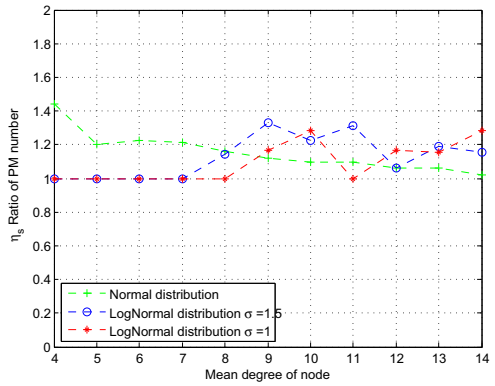

Fig. 9. The PM ratio when the mean degree of VMs ranges from 4 to 14 stepped by 1, and the number of VMs is 200.

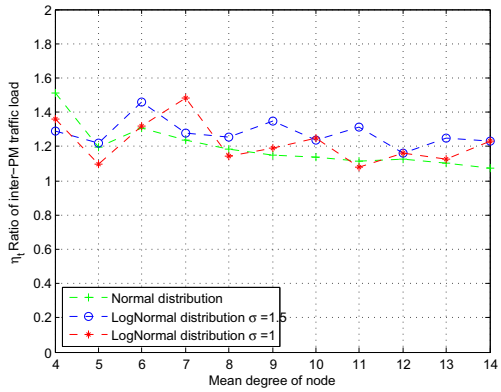

Fig. 10. The inter-PM traffic ratio when the mean degree of VMs ranges from 4 to 14 stepped by 1, and the number of VMs is 200.

First, we simulate the number of VMs of a virtual network. Fig. 7 and 8 show the PM ratio and inter-PM traffic ratio vary with the number of VMs which ranges from 50 to 600 with step size 25. The mean degree of VMs is 8. Three curves represent three edge distributions. From the figures, we observe the PM ratios η_s and inter-PM traffic ratios η_t are no less than 1 which demonstrates that Algorithm VM-Packing partitions the set of VMs into fewer PMs with fewer inter-PM traffics than Algorithm VMP does in all three cases. Compared to Algorithm VMP, Algorithm VM-Packing reduces the number of PMs by up to 20% when the number of VMs is less than 300 and by 20% to 40% when the number of VMs is more than 300 for the log-normal distribution with $\sigma = 1.5$. Further, it reduces the inter-PM traffics by 10% to 20% when the number of VMs is less than 300 and by 30% to 45% when the number of VMs is more than 300 for the log-normal distribution with $\sigma = 1.5$.

Algorithm VM-Packing is more efficient in log-normal distributions with $\sigma = 1.5$ than $\sigma = 1$, and more efficient in log-normal distributions than the normal distribution when the number of VMs is more than 300. From Fig. 7 and 8, we also observe that when the number of VMs is small, the performances of both algorithms are almost the same in the log-normal distribution with $\sigma = 1$. Occasionally, Algorithm VMP is more efficient than Algorithm VM-Packing, because Algorithm VM-Packing is more suitable in communication-intensive cases where the traffic patterns between VMs are more distinct.

Furthermore, we simulate the mean degree of VMs (nodes). Fig. 9 and 10 show the PM ratio and inter-PM traffic ratio vary with the mean degree of nodes which ranges from 4 to 14 step by 1, when the number of VMs is 200. The curves show that Algorithm VM-Packing outperforms Algorithm VMP by 11% to 43% in term of inter-PM traffics. Algorithm VM-Packing performs similar to Algorithm VMP when the mean degree of nodes is less than 8 since the traffic is low. When the degree of a VM increases, Algorithm VM-Packing becomes more efficient than Algorithm VMP.

VI. Conclusion

In this paper, we explore the virtual machine (VM) packing problem in cloud data centers. We generalize VM packing as a *Multi-Capacity Bin Packing problem*. Unlike bin packing, in the traffic-aware VM packing, the cumulative bandwidth demand of all VMs hosted in a server can be smaller than the sum of individual VM bandwidth demand because the communications between co-located VMs are usually through the internal channels and do not require any communication bandwidth from the server adapter. Our experimental results show the proposed solutions significantly outperform the existing VM placement solutions.

Acknowledgment

This work is partially supported by US NSF under the grant No. DGE1419280 and University of Michigan-Dearborn under an IAVS grant.

References

[1] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, 2011, pp. 242–253.

[2] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 2010 ACM Conference on Emerging Networking Experiments and Technology(CoNEXT 2010)*, 2010, p. 15.

[3] 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*, 2010, pp. 1–9.

[4] 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.

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

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

[7] S.-H. Wang, P. P.-W. Huang, C. H.-P. Wen, and L.-C. Wang, "Eqvmp: Energy-efficient and qos-aware virtual machine placement for software defined datacenter networks," in *Information Networking (ICOIN), 2014 International Conference on*. IEEE, 2014, pp. 220–225.

[8] L. Popa, P. Yalagandula, S. Banerjee, J. C. Mogul, Y. Turner, and J. R. Santos, "Elasticswitch: practical work-conserving bandwidth guarantees for cloud computing," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, 2013, pp. 351–362.

[9] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "Cloudnaas: a cloud networking platform for enterprise applications," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, p. 8.

[10] M. Qiu, Z. Ming, J. Li, K. Gai, and Z. Zong, "Phase-change memory optimization for green cloud with genetic algorithm," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3528–3540, 2015.

[11] D. S. Dias and L. H. M. Costa, "Online traffic-aware virtual machine placement in data center networks," in *Global Information Infrastructure and Networking Symposium (GIIS), 2012*. IEEE, 2012, pp. 1–8.