Hierarchical Virtual Machine Placement in Modular Data Centers

Linquan Zhang*, Xunrui Yin*, Zongpeng Li*, Chuan Wu†
*Department of Computer Science, University of Calgary, {linqzhan, xunyin, zongpeng}@ucalgary.ca
†Department of Computer Science, The University of Hong Kong, cwu@cs.hku.hk

***Abstract*—This work studies how to minimize communication cost for placing Virtual Machines (VMs) in a modular data center. We consider a number of cooperative VMs implementing the same job, with known inter-VM communication patterns. The modular data center has a two-layer network structure, where computing pods constitute basic building blocks and are connected by a core network. At the core network layer, we design spectral clustering algorithms to partition VMs into computing pods, minimizing inter-pod communication cost. We then further apply an SDP relaxation approach to decide the VM placement within each computing pod, targeting both load balancing among physical servers and inter-server communication cost minimization. Extensive simulations are conducted to validate the efficacy of the proposed hierarchical VM placement scheme.**

I. INTRODUCTION

The market of cloud computing is experiencing a rapid growth rate, as cloud services become an indispensable part of digital daily lives. As exemplified by Amazon EC2 and Windows Azure [1], a cloud provider organizes a large pool of shared resources such as CPU, memory, bandwidth and storage, to cater for the increasing yet dynamic needs of computing resource from users in a pay-as-you-go fashion.

Data centers are the critical infrastructure that physically hosts cloud computing resources and services. Traditional data centers often have a complicated structure, high operating and management costs, and poor flexibility and scalability. The *modular data center* architecture is proposed to address these drawbacks, simplifying data center management and offering high scalability [2]. A typical modular data center employs *computing pods* as a basic building block. Each pod consists of blade servers, storage area network arrays and switches [2], [3]. Data center operators can purchase additional computing pods and connect them to the core network in the data center to expand computing resources on the fly, almost in a *plug-and-play* fashion.

Virtualization technologies, such as Xen and KVM, help cloud providers allocate resources to users by packing them into virtual machines (VMs) on demand, ensuring isolation and high utilization of resources. As a key enabling technology of cloud computing, virtualization also leads to a number of challenges. For instance, recent studies have focused on i) the impact of virtualization overhead [4]; ii) efficient allocation mechanisms for VMs [5], [6]; iii) energy efficiency and VM consolidation [7]. In this work, we study cost-aware VM placement under load balancing consideration in a cloud, leveraging algorithmic tools from spectral graph theory and Semidefinite Programming (SDP).

A cloud user often needs dozens of VMs for concurrent threads (*e.g.*, front end, central processing, database) to work in collaboration on a sophisticated computing task (*e.g.*, online games [8], enterprise applications [9], and MapReduce applications [10]). Inter-VM communication is often important for such collaborative job execution. The quality of inter-VM communication may have a significant impact on Quality of Service and user experience, depending on the concrete deployment of VMs and topology of the data center. The communication cost of a link in the data center depends on the aggregate traffic transmitted through that link as well as the latency of the link. For example, links at the core network layer have higher latency than those within one computing pod. Minimizing the total communication cost helps improve throughput and reduce latency. Our first goal in VM placement is inter-VM communication cost minimization.

Furthermore, given a fixed number of servers, balanced placement of VMs among physical servers is desirable, since it can improve resource utilization and VM performance. Balanced placement does not contradict the VM consolidation consideration from the energy perspective. Because only a few machines, not all machines, are turned on to balance the VM workload. Carefully choosing a number of machines that are ON can improve both resource utilization and computing performance. A simple solution that places all VMs into one server may minimize the communication cost among VMs, but can also saturate that machine, making all VMs extremely slow due to the limited resource capacity of a single server in the modular data center. Our second goal in VM placement is inter-server load balancing. Combining both goals, we focus on the problem of placing VMs most evenly into given servers while minimizing the communication cost among all VMs.

This work focuses on a two-layer data center network as shown in Fig. 1. Computing pods are inter-connected through the core network layer. Inter-pod traffic cost at the core network is higher than the intra-pod counterpart. A large-scale computing task may involve collaboration among tens of even hundreds of VM instances [10], and a single physical machine or even a computing pod is not feasible in hosting all of them. VMs are hence to be distributed across the data center. We approach such VM placement in two steps. First, a tailored VM placement scheme based on *spectral graph theory* is designed for distributing VMs into different computing pods,

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.32

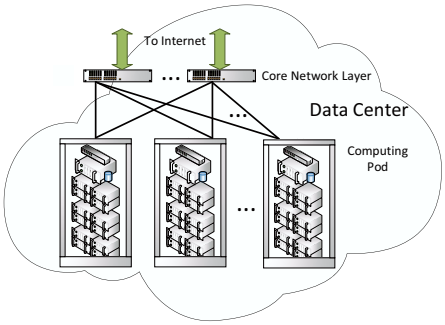


Fig. 1. An illustration of a two-layer network in a modular data center consisting of computing pods.

while minimizing the communication cost among computing pods. Eigenvalues and eigenvectors of the VM communication graph, which is an abstraction of the inter-VM traffic pattern, are calculated and utilized to partition VM instances.

The second step focuses on intra-pod VM placement. We aim to minimize inter-server communication cost among VM instances in the same pod, while simultaneously balancing the VM loads distributed to each server. We propose a transformation that converts a weighted VM into a subgraph of unit-weight VMs. An SDP based approach is then introduced to capture the essence of the underlying optimization, yielding a rough solution which will be further refined via a randomized algorithm. The SDP has an exponential number of constraints, and we propose two alternative efficient solutions to it. The first employs the ellipsoid method to solve the SDP in polynomial time, aided by a separation oracle. The second method reduces the number of constraints from exponential to polynomial by leveraging optimization results from the spectral partitioning algorithm in the first layer. The resulting algorithm is rigorously analyzed, and is shown to guarantee desired load balancing with provable approximation in communication cost minimization. Extensive simulation studies are conducted to evaluate the efficacy of the algorithms. While verifying inter-server load balancing, they also demonstrate up to 80% cost savings compared with a randomized partitioning benchmark algorithm, and up to 40% cost savings compared with an existing partition algorithm [11].

In the rest of the paper, we discuss related work in Sec. II, and introduce the system model in Sec. III. Sec. IV and Sec. V present the two-layer VM placement algorithms. Sec. VI presents simulation studies, and Sec. VII concludes the paper.

II. Previous Literature

Biran *et al.* [12] study the problem of placing VMs into physical hosts, with consideration of traffic demands, CPU and memory requirements. A heuristic algorithm is presented, without theoretical performance guarantee. Jiang *et al.* [13] study the joint VM placement and routing problem in data centers, and propose an online algorithm for handling dynamic traffic and requests. Meng *et al.*, [11] study a problem of minimizing total communication distance among VMs. They improve the network scalability by using traffic-aware VM placement. A placement system, Volley [14], is proposed to handle the VM placement problem in geo-distributed data centers. Jayasinghe *et al.* [15] study structural constraint-aware VM placement for IaaS Clouds, proposing heuristic algorithms without theoretical performance guarantee. Our VM placement problem has a two-layer structure, and our algorithm design techniques (spectral graph theory and SDP relaxation) are different from these previous studies.

Hajjat *et al.* [9] study migrating enterprise applications to cloud platforms. Due to privacy and security concerns, only some insensitive applications are hosted in cloud while others have to be kept in local servers. However their solution does not consider the inner structure of the cloud. Pujol *et al.* [16] propose a social partitioning and replication middle-ware helping scaling online social networks. Yet they do not take the structure of the data center into consideration either.

A substantial body of literature exists on network architecture design in modular data centers. For example, BCube [17] is proposed to improve the network performance inside a container, which is a building block for a modular data center. MDCube [18] aims at enhancing the performance of the interconnection among these containers.

Using eigenvalues of a graph's representation matrix for graph partitioning has been studied in theoretical computer science [19]. Such spectral partitioning (the eigenvalues form the *spectrum* of the graph) is relatively easy to implement, and may result in high quality partitions in practice, justifying a computation overhead in finding the eigenvalues and eigenvectors [20]. The eigenvalue approach pays little attention to balanced partitioning, focusing on merely cut minimization. Van Driessche *et al.* [21] adopt spectral partitioning to balance workload among multiple processors and to minimize inter-processor communication. This work resorts to SDP instead of spectral graph theory for load balancing.

III. System Model and Preliminaries

Consider a modular data center with computing pods each consisting of a number of servers. Let $\mathcal{S} = \{s_1, ..., s_M\}$ be the set of servers in the data center, where s_i is the i-th server. P_j denotes the j-th computing pod. We use a subset $\mathcal{S}' \subset \cup_{j=1}^{m} P_j \subseteq \mathcal{S}$, *i.e.*, servers in m computing pods, to serve all VM demands of the user. Let $G = (V, E)$ be an undirected *communication graph* representing a computing and communication structure with $|V|$ VMs in a cloud. Each vertex represents a VM. There is an edge $e_{uv} = (u, v) \in E$ if VMs u and v communicate with each other. Let w_{uv} represent the traffic volume between VMs u and v ($w_{uv} = 0$ if VM u does not communicate with VM v), obtained from statistical data or empirical estimation. The communication cost is linear in w_{uv}, is $\beta_p w_{uv}$ within one computing pod, and is $\beta_d w_{uv}$ across computing pods. We have $\beta_d > \beta_p$. Furthermore, when two VMs reside on the same physical server, communication cost between them is negligible and is assumed as zero.

VM instances in the cloud are heterogeneous in nature [1], [5]. VM i has an integer weight c_i, abstracted from its configurations. A large c_i represents a high-end VM consuming

a large amount of resources. A *unit VM* has a weight of 1. Suppose we want to partition the VMs into m computing pods, $P_1, \cdots, P_m$, among which the communication cost is to be minimized. We later discuss how to choose an appropriate value for m in Sec. IV. Let V_j be the set of VMs assigned to pod P_j. We further divide VMs V_j in P_j to its physical servers such that each server contains less than $(1+\epsilon)\sum_{i\in V_j} c_i/m_j$ VMs. Here $\epsilon \geq 0$; m_j is the number of physical servers available to host the VMs, and can be determined by i) capacity of each server, ii) total weighted sum $\sum_{i\in V_j} c_i$ in pod P_j. Carefully chosen m_j can improve both resource utilization and VM performance.

A binary variable x_u^s is introduced to indicate whether VM u is placed into server s or not. The communication cost among computing pods C_{ex} is:

$$C_{ex} = \sum_{s\in P_j \neq P_{j'} \ni s'} \sum_{(u,v)\in E} \beta_d w_{uv} x_u^s x_v^{s'}/2 \tag{1}$$

The term is divided by 2 since each edge is counted twice. Similarly, the communication cost within one pod C_{in} is:

$$C_{in} = \sum_j \sum_{s,s'\in P_j, s\neq s'} \sum_{(u,v)\in E} \beta_p w_{uv} x_u^s x_v^{s'}/2 \tag{2}$$

Targeting a balanced placement within a computing pod, we put at most $(1+\epsilon)\sum_{i\in V_j} c_i/m_j$ VMs into one server in pod P_j. This leads to the following balancing constraint:

$$\sum_{u\in V_j} c_u x_u^s \leq (1+\epsilon)\sum_{i\in V_j} c_i/m_j, \forall s \in \mathcal{S}' \tag{3}$$

To ensure that a VM is placed in one server only, we have:

$$\sum_{s\in\mathcal{S}'} x_u^s = 1, \forall u \in V \tag{4}$$

In conclusion, the optimization problem is shown as follows:

$$\begin{aligned} &\text{minimize} \quad C_{ex} + C_{in} \\ &\text{subject to } (3), (4) \text{ and } x_u^s \in \{0,1\}, \forall u \in V, s \in \mathcal{S}'. \end{aligned} \tag{5}$$

Since the graph partitioning problem is NP-hard [22], we resort to efficient approximation algorithms for (5).

IV. A Spectral Partitioning Algorithm for the First Layer

We now study how to place VMs into different computing pods while minimizing traffic in the core network layer. We first review basic concepts from spectral graph theory.

The *Adjacency matrix* $A = (a_{uv})$ for a weighted graph is defined as:

$$a_{uv} = \begin{cases} w_{uv} & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The *degree matrix* $D = (d_{uv})$ for a graph is defined as:

$$d_{uv} = \begin{cases} \sum_{z\in V} w_{uz} & \text{if } u = v \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The *Laplacian matrix* $L = (l_{uv})$ is defined as $L = D - A$. By definition, the Laplacian matrix is a symmetric, positive semi-definite matrix. We sort all eigenvalues of L in non-decreasing order $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_{|V|}$. $\lambda_1 = 0$ as $L\mathbf{1} = 0\mathbf{1}$ and L is positive semi-definite.

Algorithm 1 A Bisection Spectral Partitioning

```
P_1, P_2 = ∅;
Let λ_1 ≤ λ_2 ≤ ··· ≤ λ_n be the eigenvalues to L;
Pick v, the eigenvector corresponding to λ_2;
c = the median of all entries in v;
for all i ∈ {1, ··· , n} do
    if v_i > c then
        P_1 = P_1 ∪ {i};
    else
        P_2 = P_2 ∪ {i};
Output the bisection partitioning solution (P_1, P_2);
```

A. A Bisection VM Placement using λ_2

To familiarize readers with spectral partitioning, we first introduce a simple bisection approach using the second-smallest eigenvalue λ_2, known as the *algebraic connectivity* of a graph [23]. A larger λ_2 implies better connectivity of the graph. The Spectral Partitioning Algorithm as shown in Algorithm 1 employs the eigenvector $\boldsymbol{v}$ corresponding to λ_2, known as the *Fiedler vector*, to partition a given graph into two partitions. The median c of all entries in $\boldsymbol{v}$ is identified. For each entry v_i in $\boldsymbol{v}$, if $v_i > c$, vertex i is assigned to partition P_1; otherwise it is assigned to partition P_2. Consequently, a balanced partitioning is generated. The bisection process can be recursively applied to each partition until reaching a desired number of partitions.

The underlying intuition for such bisection is the following. Let $\boldsymbol{x} \in \{1,-1\}^n$. If $x_i = 1$ then vertex i is assigned to partition P_1; otherwise it is assigned to partition P_2. Then $\frac{1}{4}\boldsymbol{x}^T L\boldsymbol{x}$ is the communication cost between P_1 and P_2.

Theorem 1 (Courant-Fischer theorem [24])**.** *Let A be an $n\times n$ symmetric matrix and let $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$ be the eigenvalues of A with $\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots \boldsymbol{v}_n$ as corresponding eigenvectors, then*

$$\lambda_1 = \min_{\boldsymbol{x}\neq 0} \frac{\boldsymbol{x}^T A\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$$

$$\lambda_2 = \min_{\boldsymbol{x}\neq 0, \boldsymbol{x}\perp\boldsymbol{v}_1} \frac{\boldsymbol{x}^T A\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$$

$$\lambda_n = \max_{\boldsymbol{x}\neq 0} \frac{\boldsymbol{x}^T A\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$$

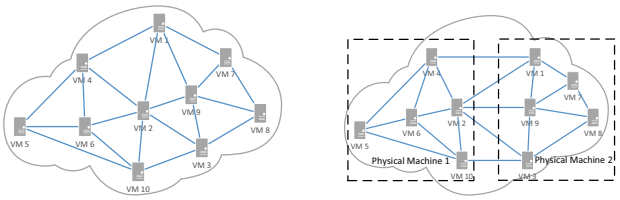
Since $\boldsymbol{x}^T\boldsymbol{x} = n$, *i.e.* $\|\boldsymbol{x}\|^2 = n$, following Courant-Fischer theorem, we know that:

$$\lambda_2 = \min_{\boldsymbol{x}\neq 0, \boldsymbol{x}\perp\boldsymbol{v}_1} \frac{\boldsymbol{x}^T L\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} = \frac{1}{n} \min_{\boldsymbol{x}\neq 0, \boldsymbol{x}\perp\boldsymbol{v}_1} \boldsymbol{x}^T L\boldsymbol{x}$$

where $\boldsymbol{v}_1$ is the eigenvector corresponding to λ_1. This implies that the eigenvector corresponding to the second-smallest eigenvalue λ_2 can minimize the communication cost between the two partitions.

The analysis above assumes the eigenvector $\boldsymbol{x}$ satisfies $\boldsymbol{x} \in \{1, -1\}^n$ and $\|\boldsymbol{x}\|^2 = n$, which are not always true for all Fiedler vectors. The spectral graph approach generates a best-effort and sub-optimal partitioning.

Simulation Illustration. We conduct a simple simulation to illustrate this approach, based on a VM communication graph among 10 VMs shown in Fig. 2(a). The communication cost between any two VMs is 1. We assume that each VM is a unit VM. We derive the Laplacian matrix and obtain the second smallest eigenvalue and its corresponding eigenvector. We use the median in the eigenvector to partition the communication graph into two parts as shown in Fig. 2(a) and Fig. 2(b).



(a) Original Graph (b) Partitioned Graph

Fig. 2. Using the second smallest eigenvector to partition the VM communication graph into two equal-size components.

B. Multi-Way VM Placement

VM Placement via Multiple Eigenvectors. Similar to the bisection process using the second-smallest eigenvalue, multiple eigenvalues can be used towards multiple-way partitioning. To divide all VMs into m computing pods, we choose the m-smallest eigenvalues of the Laplacian matrix L, *i.e.*, $\lambda_1, \lambda_2, \cdots, \lambda_m$. Using their corresponding eigenvectors, $\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^m$, we form a new $n \times m$ matrix $U = [\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^m]$. Embed these n VMs into m-dimensional Euclidean space by choosing each row of U as a set of coordinates for a point. We next apply the k-means algorithm to these n points to partition them into m parts. Algorithm 2 summarizes our multi-way VM partitioning algorithm.

Algorithm 2 A Multi-way Spectral Partitioning for VMs

1: //Choose an appropriate m, the number of computing pods
2: Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues to L;
3: Check available number of computing pods m_1;
4: Find an m_2 so that $\lambda_1, \cdots, \lambda_{m_2}$ are small while λ_{m_2+1} is relatively large;
5: $m = \min\{m_1, m_2\}$;
6:
7: //Partition the VMs into m parts
8: Pick $\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^m$, the eigenvectors corresponding to $\lambda_1, \lambda_2, \cdots, \lambda_m$;
9: $U = [\boldsymbol{v}^1, \boldsymbol{v}^2, \cdots, \boldsymbol{v}^m]$ using the eigenvectors as columns;
10: Select each column in $Y = U^T = [Y_1, Y_2, \cdots, Y_n]$ as a point;
11: Apply the k-means algorithm to partition these n points into m parts;
12: Output the m-partitioning solution;

How Many Computing Pods (Partitions) and How Many Dimensions? Since we adopt the k-means algorithm, whose performance is sensitive to the target number of partitions, we have to carefully choose m to achieve low communication cost. The best number of computing pods is decided by several factors. First, the total number of computing pods that are available. The cloud provider customarily turns some computing pods off during low-demand periods, *e.g.*, midnight, to reduce energy consumption. Second, the available capacity at the computing pod. The pod may already host other users' VMs, hence the available capacities of pods are heterogeneous.

Besides these aforementioned resource capacity constraints, the structure of the communication graph has a considerable impact on pods sizing. For a given graph $G = (V, E)$, a subset $S \subseteq V$, the *conductance* $\phi(S)$ of S is defined as

$$\phi(S) = \frac{|E(S, \bar{S})|}{\min\{|D(S)|, |D(\bar{S})|\}} \tag{8}$$

where $\bar{S}$ is the complement of S, $E(S, \bar{S})$ is the set of edges from S to $\bar{S}$, and $D(S)$ is the sum of degrees for nodes in S. Then the following higher-order Cheeger's inequalities [25] provides us useful insight on how many pods are suitable:

Theorem 2. *Given a graph $G = (V, E)$, and $k \in \mathbb{N}$, then*

$$\lambda_k/2 \leq \phi_k(G) \leq O(k^2)\sqrt{\lambda_k}$$

where $\phi_k(G) = \min_{S_1, \cdots, S_k} \max_{1 \leq i \leq k} \phi(S_i)$. $S_1, \cdots, S_k$ are non-empty, disjoint subsets of V.

A small $\phi_k(G)$ implies that the graph can be divided into k parts with a small number of cuts. Theorem 2 suggests that if $\lambda_1, \cdots, \lambda_m$ are rather small, while λ_{m+1} is substantially larger, then we can choose m as the number of computing pods. This is because dividing VMs into m pods will lead to a small number of cuts, *i.e.*, relatively low communication cost while $m + 1$ pods will incur a high cost in communication. We summarize this rule in Algorithm 2. As a special case, if $\lambda_1 = \lambda_2 = \cdots = \lambda_m = 0, \lambda_{m+1} > 0$, then according to Theorem 2, this implies that $\phi_1(G) = \cdots = \phi_m(G) = 0$, and further suggests that there exist m disjoint partitions in the graph already that can be exploited.

The dimensions used for representing each VMs in k-means clustering have a considerable impact on the performance of the partitioning. Optimally solving the k-means problem with n vertices incurs $O(n^{dm+1} \log n)$ time complexity [26], where m is the number of partitions and d is the number of dimensions. Large number of dimensions would lead to high computation complexity. Low dimensions would degrade the quality of the partitioning, as we observed in Sec. VI. We choose $d = m$ as a trade-off.

V. An SDP Algorithm for Intra-Pod VM Placement

Next we study how to place VMs into physical servers within one computing pod. Comparing the capacity of a physical machine with total VM demand, we assume that the cloud provide would allocate a bunch of vacant identical physical machines in each pod to cater for VM requests. The two

objectives for VM placement here are: (i) Minimizing inter-server communication cost within the pod, and (ii) Balancing the (weighted) number of VMs placed in different servers.

Different from the spectral clustering problem in Sec. IV, here we need to ensure balanced VM placement across physical machines, besides communication cost minimization. Using the Fiedler vector to recursively separate a graph into two partitions could provide a balanced VM placement. However such recursive partitioning provides no guarantee on approximation ratio, as compared to optimal partitioning.

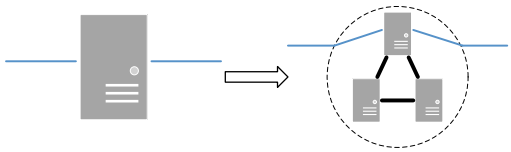


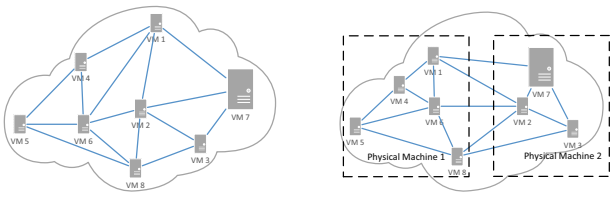
Fig. 3. Coverting a VM whose weight is 3 to three unit VMs. Communication cost among these three unit VMs is H and is very high.

We first describe a procedure that translates the problem of placing heterogeneous VMs into the problem of placing uniform VMs, and assume uniform VMs subsequently.

Placement of Heterogeneous VMs.

As illustrated in Fig. 3, given a VM u and its weight c_u, we construct a subgraph containing c_u unit VMs, each of which connects to the other $c_u - 1$ unit VMs with a sufficiently high communication cost H. We then arbitrarily pick one of these c_u unit VMs, u', and reconnect the communication links, which are originally connect to VM u, to the unit VM u'.

The c_u-clique resulting from the conversion above should not be separated (physically infeasible) during the VM placement procedure, which can be guaranteed by setting the communication cost H among these unit VMs to a sufficiently high value. Fig. 4(a) and Fig. 4(b) provide an example of heterogeneous VM placement.



(a) Original Graph (b) Partitioned Graph

Fig. 4. Placing weighted VMs into two servers, where the weight of VM 7 is 3 and other VMs are of unit-weight.

Randomized VM Placement within One Pod. After the conversion, we obtain a new subgraph (V_j', E_j') where V_j' represents all unit VMs, E_j' is the set of communication edges whose two VMs are both in V_j', .

Let $n' = |V_j'|$. Let $\boldsymbol{v}$ be a vector in $\mathbb{R}^{n'}$, associated with VM $v \in V_j'$. We formulate an SDP to capture the intra-pod VM placement optimization:

$$\text{minimize} \sum_{(u,v)\in E_j'} \frac{1}{2} w'_{u,v} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \tag{9}$$

subject to:

$$\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \|\boldsymbol{v} - \boldsymbol{z}\|_2^2 \geq \|\boldsymbol{u} - \boldsymbol{z}\|_2^2 \qquad \forall u, v, z \in V_j' \tag{9a}$$

$$\sum_{v\in S} \frac{1}{2} \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \geq |S| - n'/m_j \qquad \forall u \in S, S \subset V_j' \tag{9b}$$

$$\|\boldsymbol{u}\|_2^2 = 1 \qquad \forall u \in V_j' \tag{9c}$$

The above SDP essentially projects the VM instances to a hyper-sphere surface for partitioning into physical servers. Constraint $(9a)$ expresses triangular inequality in the geometric space. Constraint $(9b)$ makes sure that VMs are not too close to each other. Constraint $(9c)$ ensures that all VMs spread over the surface of a unit sphere. While SDP belongs to convex optimization and has general solution techniques such as the interior-point algorithm, the SDP here has an exponential number of constraints and requires special solution techniques. Below we discuss two alternative solutions that both run in polynomial time.

We propose a heuristic solution to the SDP that achieves polynomial time complexity by first reducing the number of constraints in $(9b)$ from exponential to polynomial, and then applying an interior-point solution algorithm. Recall that the spectral partitioning algorithm described in Sec. IV.B embeds the n' VMs into an m_j-dimension Euclidean space, based on which the k-means clustering algorithm generates a good (but possibly unbalanced) partition. We refer to this embedding to enforce a restriction on the order of the squared distances $\|\boldsymbol{u}-\boldsymbol{v}\|_2^2$, with which $(9b)$ can be simplified into $n'(2n'-3) \in O(n'^2)$ constraints.

Specifically, let $\boldsymbol{u}'$ be the coordinates of node u in the embedding generated by the spectral partitioning algorithm. We replace $(9b)$ with the following two sets of constraints: for each $u \in V_j'$,

$$\|\boldsymbol{u} - \boldsymbol{v}_i\|_2^2 \leq \|\boldsymbol{u} - \boldsymbol{v}_{i+1}\|_2^2, \quad 1 \leq i \leq n' - 2$$

$$\sum_{i=1}^{l-1} \tfrac{1}{2} \|\boldsymbol{u} - \boldsymbol{v}_i\|_2^2 \geq l - n'/m_j, \quad l = 2, 3, \cdots n'$$

where $v_1, v_2, \cdots, v_{n'-1}$ is a rearrangement of nodes $V_j' \backslash \{u\}$ such that $\|\boldsymbol{u}' - \boldsymbol{v}_1'\|_2^2 \leq \|\boldsymbol{u}' - \boldsymbol{v}_2'\|_2^2 \leq \cdots \leq \|\boldsymbol{u}' - \boldsymbol{v}_{n'-1}'\|_2^2$.

Algorithm 3 A Heuristic Partitioning Algorithm for Intra-Pod VM Placement

1: //Assume weighted VMs have been converted into unit VMs and $\{\boldsymbol{u}, u \in V_j'\}$ is the optimal solution of SDP (5)
2: Call Algorithm 4 to obtain solution C;
3: **if** $\epsilon \geq 1$ **then**
4: Call Algorithm 5 to obtain solution C';
5: Replace C with C' if C' has a smaller cost;

Given the SDP solution method above, we first solve SDP (9), obtaining a relaxed solution to intra-pod VM placement in polynomial time. This solution acts as a lower bound on the cost of an optimal solution to the m_j-partitioning problem. We then use the randomized partitioning algorithm in Algorithm 5 to generate each server partition. The high level picture

Algorithm 4 A Heuristic SDP Rounding Algorithm

1: $C = \emptyset$;
2: **while** $|V| > (1+\epsilon)n'/m_j$ **do**
3: **for all** $u \in V$ **do**
4: $S(u) = \{v \in V \mid \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \leq \frac{\epsilon}{1+\epsilon}\}$;
5: $u_0 = \arg\max\{|S(u)|, u \in V\}$;
6: $C = C \cup \{S(u_0)\}$;
7: $V = V \setminus S(u_0)$;
8: **if** $|V| > 0$ **then**
9: $C = C \cup \{V\}$;
10: **while** $|C| > m_j$ **do**
11: Let $C_1 \in C$ be the smallest partition
12: **for all** $u \in C_1$ **do**
13: $i = \arg\max\{w(u, C_i), C_i \in C$ and $|C_i| + 1 \leq (1+\epsilon)n'/m_j\}$
14: //where $w(u, C_i) = \sum_{v \in C_i} w(u, v)$
15: $C_i = C_i \cup \{u\}$;
16: $C = C \setminus \{C_1\}$;

Algorithm 5 A Randomized Partitioning Algorithm for $\epsilon \geq 1$

1: $C = \emptyset$;
2: ν is a value such that $Pr(X \geq \nu) = 2m_j/\epsilon$, where $X \sim N(0,1)$.
3: **while** $|V| > (1+\epsilon)n'/m_j$ **do**
4: Generate a random vector $\boldsymbol{r}$, each element of which is generated from normal distribution $N(0,1)$;
5: $V_{tmp} = \{v | v \in V, g(v) \cdot \boldsymbol{r} \geq \nu\}$;
6: **if** $|V_{tmp}| \in (0, (1+\epsilon)\frac{n'}{m_j})$ **then**
7: $C = C \cup \{V_{tmp}\}$;
8: $V = V \setminus V_{tmp}$;
9: **if** $|V| > 0$ **then**
10: $C = C \cup \{V\}$;

of Algorithm 3 is as follows. Subroutine Algorithm 4 and Algorithm 5 are designed to handle the two cases $\epsilon < 1$ and $\epsilon \geq 1$, respectively. For the case $\epsilon < 1$, we propose Algorithm 4, which is a heuristic SDP rounding algorithm. The basic ideas are: with an SDP solution, the set of nodes near a node $u \in V_j'$ cannot be too large. Specifically, the set $S(u) = \{v \in V \mid \|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \leq \frac{\epsilon}{1+\epsilon}\}\}$ is of size no more than $(1+\epsilon)n'/m_j$ [27]. The SDP relaxation ensures that the rounding subroutine generates an optimal solution if the number of partitions is not restricted. If there are more than m_j partitions after the rounding subroutine, we merge small partitions into larger ones according to the link weights.

For the case $\epsilon \geq 1$, we run both subroutines, select the one with minimum communication cost by comparing the two results. Subroutine Algorithm 5 works as follows: randomly choose a vector and assign all vertices that are close to this vector to a new potential cluster. If the selected cluster is empty or too crowded, then pick a new randomized vector and choose a new cluster again until a good cluster is found. In Algorithm 5, each component is at most $(1+\epsilon)n'/m_j$ when $\epsilon \geq 1$. The function $g(\boldsymbol{v})$ defined in line 9 is a transformation that converts distances in l_2^2 on $\mathbb{R}^d$ to distances in $\mathbb{R}^{n'}$ Euclidean space. If we obtain all vertices $\boldsymbol{v} \in \mathbb{R}^d = \mathbb{R}^{n'}$ by solving the SDP relaxation, then we can simply set $g(\boldsymbol{v}) = \boldsymbol{v}$. In simulation studies, we observe that $d \leq n'$ for the solution to SDP (9).

The following theorem provides a performance guarantee that the placement into physical machines will not incur a high cost in inter-server VM communication.

Theorem 3. *Given $\epsilon \geq 1$, for the subgraph (V_j', E_j') in the computing pod P_j, Algorithm 3 can produce m_j partitions each of which is at most $(1+\epsilon)n'/m_j$ in size, in expected polynomial time with expected communication cost at most $O(\sqrt{\log n' \log m_j})$ times the optimum.*

Proof: Subroutine Algorithm 5 may produce more than m_j partitions, but the following lemma shows that they can be converted to no more than m_j partitions while the number of VMs in each partition is less than $(1+\epsilon)n'/m_j$ without increasing the communication cost.

Lemma 1. *For $\epsilon \geq 1$ and the subgraph (V_j', E_j') in the computing pod P_j, if Algorithm 5 produces a solution with more than m_j partitions, then we can merge the two smallest partitions together recursively to covert the solution to exact m_j partitions without increasing the communication cost and violating the constraint that each partition's size is at most $(1+\epsilon)n'/m_j$.*

Proof to the Lemma: Assume Algorithm 5 produces $m_j + q$ partitions, $|C_1| \leq |C_2| \leq \cdots \leq |C_{m_j+q}|$. For the sake of contradiction, suppose the new partition created by merging the two smallest partition C_1 and C_2 is larger than $(1+\epsilon)n'/m_j$, then we have $|C_2| \geq (1+\epsilon)n'/2m_j$ which implies that

$$|C_{m_j+q}| \geq \cdots \geq |C_2| \geq (1+\epsilon)n'/2m_j$$

therefore the total size for partitions $C_2, \cdots, C_{m_j+q}$ is larger than $\frac{(1+\epsilon)n'}{2m_j}(m_j + q - 1) > n'$. It contradicts the fact that the total size is n'. Thus the new created partition is not larger than $(1+\epsilon)n'/m_j$.

Merging two partitions reduces the communication cost since it reduces the communication links across partitions. That concludes the proof to Lemma 1. Combining Lemma 1 with techniques of Krauthgamer *et al.* [27], we can verify claims in Theorem 3 regarding m_j VM groups. □

VI. Performance Evaluation

We simulate 50 VM instances in a collaborative cloud task. Each VM has a weight of 3 by default. The traffic pattern is randomly generated. We assume the modular data center consists of a hundred computing pods, each of which hosts blade servers, storage area network arrays and switches and, can host up to 100 unit VMs. Considering that some existing VMs have already occupied the physical machines, we assume that there are 50 available computing pods, each with residual capacity of 30 unit VMs. The communication cost among computing pods is higher than that within one

pod due to bandwidth, delay, as well as traffic congestion. We set $\beta_d = 10, \beta_p = 1$ accordingly.

A. VM Placement in the First Layer

We use the eigenvalue method described in Sec. IV.B to determine the computing pods used to host VMs. First we examine the eigenvalues of the Laplacian matrix of the communication graph, shown in Fig. 5 in ascending order. Since the smallest eigenvalue of a given graph is always 0, we do not consider the eigenvalue difference between λ_1 and λ_2. We observe that $\lambda_2, \cdots, \lambda_5$ are small, while λ_6 is relatively large compared with $\lambda_2, \cdots, \lambda_5$.

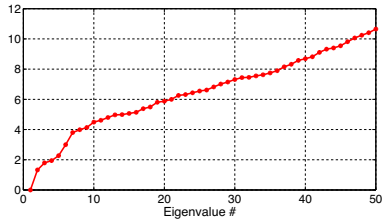


Fig. 5. The 50 eigenvalues in acceding order.

Based on the above observation, we execute Algorithm 2 to partition the VMs into 5 computing pods. Fig. 6 shows the exact placement for the first layer for various numbers of pods. The algorithm generates a rather balanced placement for 5 computing pods, in each of which 10 VMs are placed. The placement becomes rather unbalanced when the numbers of computing pods are $2, 3, 4$ and 6.

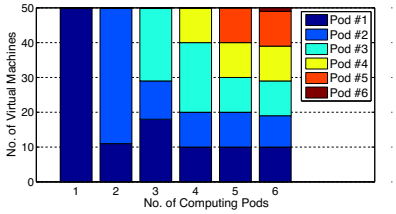


Fig. 6. The detailed partitions for different numbers of computing pods.

Fig. 7 shows the communication costs under different numbers of computing pods. Generally, the communication cost increases as the number of computing pods increases. The cost for 5 pods is rather close to that for 3 and 4 pods. This is in line with the results from the analysis of the eigenvalues. Therefore we choose $m = 5$, to divide these VMs into 5 computing pods in the first layer.

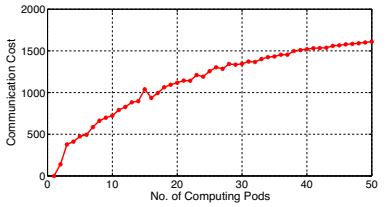


Fig. 7. Communication cost when the number of computing pods increases.

We next investigate the impact of the dimensions used for embedding VMs in Algorithm 2. The default choice is m dimensions. Fig. 8 illustrates the relation among the number of computing pods, the dimensions of points and the communication cost. Given a fixed number of computing pods, the communication cost decreases as the number of dimensions increases, *e.g.*, for the case $m = 5$, the communication cost first deceases as the number of dimensions increases, then it becomes relatively stable after 5 dimensions, with only small fluctuations.

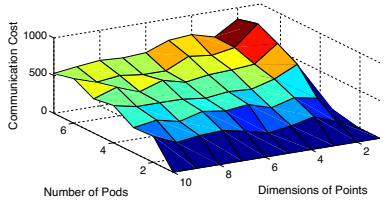


Fig. 8. Different dimensions of points in Algorithm 2 in the first layer.

B. VM Placement within Each Computing Pod

According to the first layer's partition results, we focus on the case that the VMs have been divided into 5 computing pods, and apply Algorithm 3 to further place the VMs to physical servers.

We choose $m_j = 3$ for each computing pod P_j. In the simulations, the parameter ϵ is chosen from $\{0.2, 0.5, 0.8, 1.1, 1.4\}$, which corresponds to the cases that each physical machine can hold $(1+\epsilon)n'/m_j = 4, 5, 6, 7, 8$ VMs, respectively. A partition result is shown in Fig. 9(a)-9(c) for various ϵ. We can see that the partition is fairly balanced, where only a few physical machines are assigned with the max number of VMs.

For each computing pod, we repeat Algorithm 3 for 1000 times, and show the average inner-pod communication costs in Fig. 10. We can see that the communication costs within each computing pod decrease as ϵ increases. This is because with a large ϵ, the partition algorithm is allowed to assign more VMs to one physical machine.

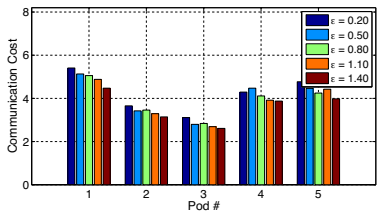
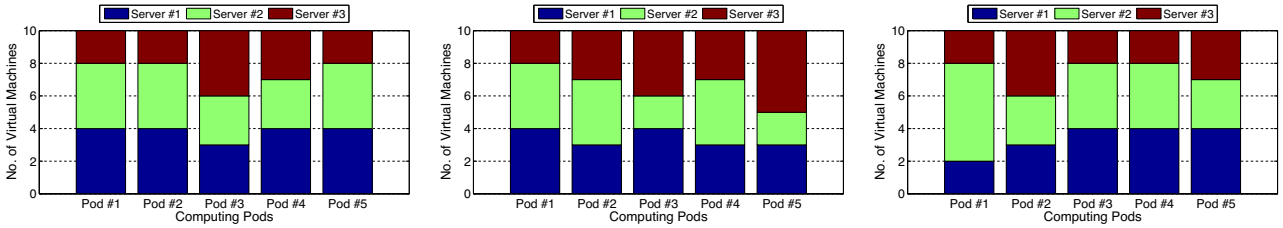


Fig. 10. The role of unbalance ratio ϵ.

We further compare Algorithm 3 with a simple partitioning algorithm and the Cluster-and-Cut algorithm proposed by Meng *et al.* [11]. The simple partitioning algorithm randomly picks a balanced partition. We also use the parameter ϵ to represent the balanced partition constraint. The Cluster-and-Cut algorithm does not aim to a balanced partition. Instead, it assumes each physical server has a constant capacity. In the experiment, we set an appropriate capacity for each server such that the Cluster-and-Cut algorithm has the same number of partitions as the others do. The communication costs are shown in Fig. 11. We can see that under the same unbalance ratio ϵ,



(a) $\epsilon = 0.2$ (b) $\epsilon = 0.5$ (c) $\epsilon = 0.8$

Fig. 9. VM placement in each computing pod.

our algorithm achieves a VM communication cost reduction of 60% – 80% compared with the simple partitioning algorithm, and 5% – 40% compared with the Cluster-and-Cut algorithm.

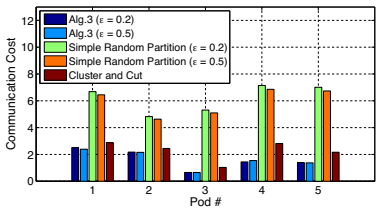


Fig. 11. Comparison of performance between Algorithm 3 and a randomized partitioning algorithm.

VII. Conclusion

This work studied VM placement for simultaneous communication cost minimization and load balancing in a modular data center, where computing pods as basic building blocks are connected by a core network. In this two-layer datacenter network, spectral clustering is employed to partition VMs into computing pods in the core network layer. Then an SDP relaxation approach is further applied to decide the VM placement within each computing pod, aiming at both load balancing across physical servers and minimizing inter-server communication cost. To our knowledge, this work is the first that conducts a systematic study of hierarchical VM placement in a modular data center.

References

[1] *Windows Azure*, http://www.microsoft.com/windowsazure/.

[2] T. Sherbak and C. Auger, "Computing Pods: Large-Scale Building Blocks for Intelligent, Automated Data Center Deployments," *Dell Power Solutions*, pp. 37 – 41, June 2009.

[3] *Cisco Virtualized Multi-Tenant Data Center, Version 2.0 Design Guide*, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/VMDC/2-0/large_pod_design_guide/vmdc20Lpdg.pdf.

[4] G. Wang and T. Ng, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center," in *Proc. IEEE INFOCOM*, 2010.

[5] L. Zhang, Z. Li, and C. Wu, "Dynamic Resource Provisioning in Cloud Computing: A Randomized Auction Approach," in *Proc. of IEEE INFOCOM*, 2014.

[6] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," in *Proc. of ACM SIGMETRICS*, 2014.

[7] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya *et al.*, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[8] RightScale, "Social Gaming in the Cloud: A Technical White Paper," *White Paper*, 2010.

[9] M. Hajjat, X. Sun, Y.-W. E. Sung, D. A. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward Bound: Planning for Beneficial Migration of Enterprise Applications to the Cloud," in *Proc. of ACM SIGCOMM*, 2010.

[10] *Amazon Elastic MapReduce*, http://aws.amazon.com/elasticmapreduce/.

[11] X. Meng, V. Pappas, and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," in *Proc. of IEEE INFOCOM*, 2010.

[12] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A Stable Network-Aware VM Placement for Cloud Systems," in *Proc. of IEEE/ACM CCGrid*, 2012.

[13] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. of IEEE INFOCOM*, 2012.

[14] S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, A. Wolman, and H. Bhogan, "Volley: Automated Data Placement for Geo-distributed Cloud Services," in *Proc. of USENIX NSDI*, 2010.

[15] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement," in *Proc. of IEEE SCC*, 2011.

[16] J. M. Pujol, V. Erramilli, G. Siganos, X. Yang, N. Laoutaris, P. Chhabra, and P. Rodriguez, "The Little Engine(s) That Could: Scaling Online Social Networks," in *Proc. of ACM SIGCOMM*, 2010.

[17] 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 *Proc. of ACM SIGCOMM*, 2009.

[18] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang, "MDCube: A High Performance Network Structure for Modular Data Center Interconnection," in *Proc. of ACM CoNEXT*, 2009.

[19] D. Spielmat and S.-H. Teng, "Spectral Partitioning Works: Planar Graphs and Finite Element Meshes," in *Proc. of IEEE FOCS*, 1996.

[20] J. Teresco, K. Devine, and J. Flaherty, "Partitioning and Dynamic Load Balancing for the Numerical Solution of Partial Differential Equations," in *Numerical Solution of Partial Differential Equations on Parallel Computers*, ser. Lecture Notes in Computational Science and Engineering, A. Bruaset and A. Tveito, Eds. Springer Berlin Heidelberg, 2006, vol. 51, pp. 55–88.

[21] R. V. Driessche and D. Roose, "An Improved Spectral Bisection Algorithm and its Application to Dynamic Load Balancing," *Parallel Computing*, vol. 21, no. 1, pp. 29 – 48, 1995.

[22] S. Arora, S. Rao, and U. Vazirani, "Expander Flows, Geometric Embeddings and Graph Partitioning," in *Proc. of ACM STOC*, 2004.

[23] M. Fiedler, "Algebraic Connectivity of Graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 98, pp. 298–305, 1973.

[24] D. Spielman, *Spectral Graph Theory*. Chapman and Hall/CRC, 2012.

[25] J. R. Lee, S. Oveis Gharan, and L. Trevisan, "Multi-way Spectral Partitioning and Higher-order Cheeger Inequalities," in *Proc. of ACM STOC*, 2012.

[26] M. Inaba, N. Katoh, and H. Imai, "Applications of Weighted Voronoi Diagrams and Randomization to Variance-based K-clustering," in *Proc. of ACM SCG*, 1994.

[27] R. Krauthgamer, J. Naor, and R. Schwartz, "Partitioning Graphs into Balanced Components," in *Proc. of ACM-SIAM SODA*, 2009.