Elasticity-aware Virtual Machine Placement for Cloud Datacenters

Kangkang Li, Jie Wu, and Adam Blaisse
Department of Computer and Information Sciences
Temple University, Philadelphia, PA, 19122
Email: {kang.kang.li, jiewu, adam.blaisse}@temple.edu

***Abstract*—With the increasing popularity of cloud computing, the cloud datacenter suffers from both limited resources and the variation of users' requests. One important feature of cloud computing is on-demand scaling, enabling the fluctuation of one user's resource demand. However, amongst previous work concerning the virtual machine (VM) placement in datacenters, satisfying the VMs' requested resources of users is the primary objective, neglecting future demand variation. In this paper, we propose the concept of *elasticity*, referring to how well the datacenter can satisfy the growth of the input VMs resource demands under both the limitations of physical machines (PMs) capacities and links capacities. To consider both dimensions of the machine and bandwidth resources simultaneously, we propose our hierarchical VM placement algorithm. We also prove the optimality of our algorithm in a frequently used semi-homogeneous datacenter configuration. Furthermore, we study the heterogeneous datacenter configuration, favoring the characteristics of multi-tenant datacenters. Evaluation results validate the efficiency of our algorithm.**

***Index Terms*—Elasticity-aware; VM placement; Datacenters**

I. Introduction

Nowadays, datacenters are becoming the mainstream hosting platform for cloud services. Due to the frequently exhibition of high link utilization [1], today's datacenter usually reserves more bandwidth for the upper layer links. With the resource limitation of both physical machines (PMs) and links, previous works mainly focus on the efficient resource management. One basic issue is the VM placement problem, which is a complicated task involving various constraints. However, among the existing literatures, most studies only consider satisfying the resource demands of VM requests, neglecting the future variation of VMs' resource demands. This is not an efficient way to manage the limited resources.

In this paper, we assume that VMs have identical machine resource demands (i.e. CPU) of R and bandwidth demands of B. Due to various reasons (e.g. incremental tasks from users), the resource demands may fluctuate. If R and B increase to R' and B', then the growth ratios of $\frac{R'-R}{R}$ and $\frac{B'-B}{B}$ describe, respectively, to what extent the growth of machine and bandwidth demands could be satisfied. So we define the *machine / bandwidth elasticity* as the largest ratio that the machine / bandwidth demand of each VM could increase. Due to the heterogenous resources, for one VM, the elasticities of machine and bandwidth resources are not the same. And we choose the smaller elasticity to be the *combinational elasticity* of the corresponding VM. Furthermore, the VMs belonging to the same user usually require the same growth ratio, while these VMs scattered across the datacenter are not likely to have the same elasticity. Therefore, we pay attention to the worst-case. That is, we select the minimal elasticity among all VMs in the datacenter as the objective to be maximized.

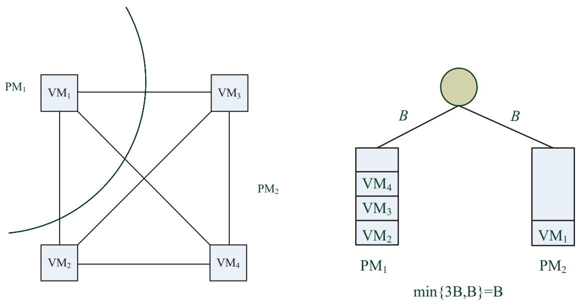


Fig. 1. Communication model

A hose model [2] is used to calculate the bandwidth demands of PMs, as shown in Fig. 1. As aforementioned, each VM desires an identical bandwidth, B, to communicate with the other VMs, while the bandwidth allocated to each pair of VMs is *uncertain*. For example, the bandwidth assigned between VM_1 and VM_2 is unknown, while the total bandwidth token by VM_1 is B. Then, PM_1 has a communication demand of *at most* B. This is because: (1) PM_1 can use at most B bandwidth since only VM_1 locates inside it; (2) VMs placed outside PM_1 (three VMs) can use at most $3B$ bandwidth; (3) the bandwidth desired by PM_1 is limited to both the VMs located inside and outside PM_1, i.e., $\min\{B, 3B\} = B$. Therefore, the bandwidth desired by a PM is the minimum bandwidth demands of the VMs located inside and outside it.

In order to maximize the combinational elasticity, we need to consider both machine elasticity and bandwidth elasticity. An example is shown in Fig. 2 (each PM has 10 VM slots, each link bandwidth is 8 Gbps). Now we want to place 10 VMs, each of which needs one VM slot in the PM and 1 Gbps bandwidth. In order to optimize the machine elasticity of each VM, we should adopt load balancing placement, and each PM is assigned 5 VMs. In that case, the maximal machine resource of each VM could increase to $\frac{10}{5} = 2$ slots (the machine elasticity is $\frac{2-1}{1} = 100\%$). However, according to the hose communication model, the bandwidth usage on the

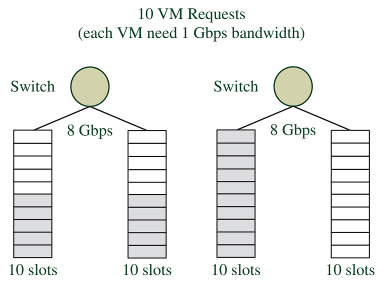


Fig. 2. Illustration of VM placement

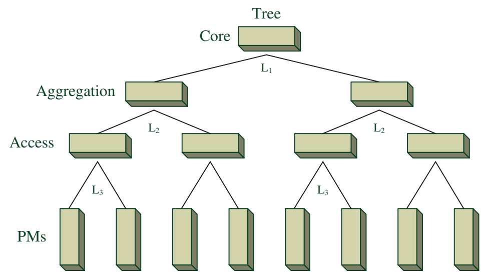


Fig. 3. Tree-based network topology

links connecting two PMs are 5 Gbps, leading to less reserved links resources. The maximal bandwidth resource of each VM could only increase to $\frac{8}{5} = 1.6$ Gbps (the bandwidth elasticity is only $\frac{1.6-1}{1} = 60\%$). On the other hand, if we want to maximize the bandwidth elasticity, we should take a load-unbalancing placement, which puts all 10 VMs on one PM. In that case, there is no bandwidth load on the link connecting two PMs. However, the machine resources are sacrificed, since the machine elasticity is 0% (each VM cannot have any growth in machine resources). From here, we could see the conflict on the optimization of machine and bandwidth elasticity.

In a multi-layer cluster with M machines and N VM requests, traversing all the possibilities to partition the N VMs into M machines could find an optimal solution; however, it would be extremely time-consuming. Therefore, we propose a hierarchical scheme that recursively places VMs step by step from the top layer to the bottom layer.

The remainder of the paper is organized as follows: In Section II, we formulate the maximal-elasticity VM placement problem. In Section III, we study a one-layer cluster with its optimal solution. Section IV focuses on the multi-layer cluster, and gives the hierarchical VM placement algorithm. In Section V, we study the heterogeneous datacenter configuration. Section VI conducts the simulations to validate the efficiency of our algorithm. In Section VII, we extend our algorithm into a K-ary tree topology with a linear elasticity relationship. In Section VIII, we introduce some previous work. Finally, conclusions are in Section IX.

II. Problem Formulation

In this section, we formulate the maximal-elasticity VM placement problem in a multi-layer binary tree datacenter. The datacenter configuration is semi-homogeneous, as shown in Fig. 3. Each PM has the same capacity of C. Also, each link of the same layer has the same bandwidth capacity: L_k (the k^{th} layer link capacity). However, the upper layer links have larger bandwidth capacities than the lower layer links, i.e., $L_1 \geq L_2 \geq L_3$. The links capacities only differ between layers, we refer to this as the *semi-homogeneous* configuration, which is widely used to ease upper-layer link congestion.

In this paper, we only study the scenario of VM requests with homogeneous resource demands. Each VM has an identical machine resource demand of R and bandwidth demand of B. The machine and bandwidth elasticities of a VM_i are denoted as E_i^r and E_i^l, with combinational elasticity to be $E_i = \min\{E_i^r, E_i^l\}$. Then, we have:

$$R_i' = R_i * (1 + E_i) \quad \text{and} \quad B_i' = B_i * (1 + E_i) \tag{1}$$

R_i' and B_i' are, respectively, the potential future machine and bandwidth resource demands of a VM_i. Our objective is to maximize the achievable E_i among all the VMs:

$$Maximize \ \min_i\{E_i\} \tag{2}$$

E_i is limited by both the PM capacities and the link capacities. First, let us consider the machine elasticity. Suppose there are m_r VMs allocated into PM_r, then the maximal potential resource is $R' = \frac{C}{m_r}$, with the machine elasticity to be $\frac{R'-R}{R}$. To maximize the machine elasticity, we have:

$$Maximize \ E^r = \min_r\{\frac{C}{R * m_r}\} \tag{3}$$

On the other hand, the bandwidth elasticity is constrained by the links. Suppose that, the sub-tree of link l with bandwidth capacity of L contains m_l VMs. According to the hose model, the bandwidth requirement for link l is $min\ B * \{m_l, N - m_l\}$. To maximize the bandwidth elasticity, we have:

$$Maximize \ E^l = \min_l\{\frac{L}{B * \min\{m_l, N - m_l\}}\} \tag{4}$$

According to Eqs. 3 and 4, the combinational elasticity is:

$$E = \min\{\min_r\{\frac{C}{R * m_r}\}, \min_l\{\frac{L}{B * \min\{m_l, N - m_l\}}\}\} \tag{5}$$

The combinational elasticity in Eq. 5 is our objective to maximize. Notably, maximizing the elasticity can be viewed as minimizing the utilization of the PM and link resources. So we transfer the objective functions Eqs. 3 and 4 into:

$$Minimize \ \max_r\{m_r \frac{R}{C}\} \tag{6}$$

$$Minimize \ \max_l\{\min\{m_l, N - m_l\}\frac{B}{L}\} \tag{7}$$

According to Eqs. 6 and 7, the combinational utilization is:

$$U = \max\{\max_r\{m_r \frac{R}{C}\}, \max_l\{\min\{m_l, N - m_l\}\frac{B}{L}\}\} \tag{8}$$

Again, maximizing the elasticity is equivalent to minimizing the utilization. In the next section, we focus on minimizing the utilization in Eq. 8. For simplicity, let each VM's machine resource demand as one VM slot [3]. Let $\langle N, B\rangle$ denote that there are N VM requests, and each VM's bandwidth is B.

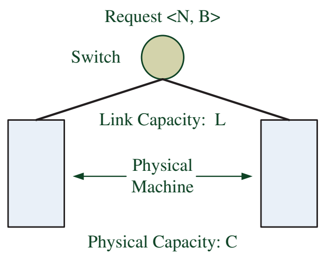


Fig. 4. One-layer cluster

III. A ONE-LEVEL CLUSTER STUDY

We start from a simple case of a one-layer cluster, which is extended into a multi-layer cluster case later. As shown in Fig. 4, we have N VMs requests with bandwidth, each with a demand of B. Each of the two PMs has a machine capacity of C, and each of the two links' capacity is L. At first, we need to determine how many VM requests could be accepted.

First, let us consider the machine resources. For each PM, the sum of the VMs' machine resources should not exceed the capacity limit. Therefore, we have $N \leq 2C$. Second, we take care of the link capacity. Suppose that each PM has 5 VM slots, each VM has a bandwidth demand of B, and the capacity of each link is $2B$. If we allocate one PM with 5 VMs, based on the hose communication model, the other PM could maximally hold 2 VMs due to link constraint. In the view of link capacity limit, the maximal number is $C + \frac{L}{B}$. So we have $N \leq \; min\{2C, C + \frac{L}{B}\}$, which is the maximal number of VMs that a one-level cluster can support. Hence, given N VM requests, the number of VMs that can be accepted is:

$$N^* = min\{N, 2C, C + \frac{L}{B}\} \tag{9}$$

A. Elasticity Maximization in One-level Clusters

There are two main factors that determines the elasticity. The first one is the input size. With limited resources, large inputs will consume more resources, leading to worse elasticity. The second is the VM placement, which is our major concern. With N VM requests, there are N^* VMs accepted into the cluster (Eq. 9). Assume that we place x VMs in the left machine, and leave the $N^* - x$ to the right node (without loss of generality, let $x \leq N^* - x$). As the value of x increases from 0 to $\lfloor \frac{N^*}{2} \rfloor$, due to the symmetry of binary-tree, there are $\lfloor \frac{N^*}{2} \rfloor + 1$ different ways to allocate the VMs in two machines. However, the value of x may not be able to achieve 0 due to the limitations of PM's capacity, or $\lfloor \frac{N^*}{2} \rfloor$ due to the limitations of the links capacity. As x increases, the utilization decreases, since the input VMs are allocated in a more balanced manner. However, the utilization of the links increases, since the traffic between the VMs in two PMs are more heavily loaded.

In this one-level cluster, links have identical capacities; we can transfer the utilization in Eq. 7 of the link as $x * \frac{B}{L}$. Then, utilization of the machine resources in Eq. 6 could be transferred as $\max\{\frac{N-x}{C}, \frac{x}{C}\} = \frac{N-x}{C}$. Then, the combinational utilization of PMs and links in Eq. 8 is:

$$U(x) = \max \; \{x\frac{B}{L}, \frac{N-x}{C}\} \tag{10}$$

The optimal solution to minimize $U(x)$ is in Appendix A.

B. Discussion

As we can see from Eq. 18, when $BC \leq L$, the optimal solution for x is $\lfloor \frac{N^*}{2} \rfloor$, i.e., load-balancing. The insight is that, the machine resource is not opulent compared to the link resources. We refer this case as the machine resources hungry, i.e., the machine resources dominate. When $BC \geq L$, the link resources are comparatively scarce. Load balancing will cause large consumption on the links. To favor the link, we have to use load-unbalancing. At this time, the optimal solution is no longer evenly divided. If all the VMs are allocated into one machine, the link capacity is not used at all. The more scarce the link capacity is, the more load-unbalancing is needed.

IV. MULTI-LAYER CLUSTER STUDY

It is time-consuming to exhaustively search for the optimal solution. However, based on the optimal results of the one-layer cluster, we can generalize this solution to multi-layer clusters, which has a low time complexity,

A. Binary Abstraction

For each switch in each layer, we can view its sub-trees as abstraction nodes. Then, the multi-layer cluster could be abstracted as a one-level binary cluster, which is easier to study. However, the obstacle is to figure out how to abstract the left or right sub-trees into a single abstraction node. We need to determine the accumulative capacity of the abstraction node. Since a sub-tree consists of constraints of both links and bottom-layer machines, we cannot just add up all the machine resources of the machines as the accumulative capacity, especially when the cluster is link resource hungry. The accumulative capacity needs to reflect resource constraints of both the links and PMs.

In order to combine the bandwidth and machine resources, we must first unify the measuring unit. For the machine resources, the measurement unit is the VM slot. Each PM has several slots, reflecting how many VMs it can support. For the bandwidth resources of links, we also want to convert them into VM slots. We suppose that a VM has a bandwidth demand of B, and the link capacity is $2B$. Based on the hose model, this link could support the communication of two VM slots, i.e., the measuring unit can be converted into VM slots. Therefore, we can use VM slots as the measuring unit to represent the accumulative capacity of an abstraction node.

We can view each switch as the root of a one-layer binary cluster, and try to abstract it into a single node. Starting from the bottom-layer, each access connects two identical PMs with capacities of C. On the one hand, each PM can support at most C VM slots. That is, $N \leq C$. On the other hand, constrained by link capacity, each PM under that link can support $\frac{L}{B}$ VM slots. If $L \geq BC$, then, one PM can support C VMs. However,

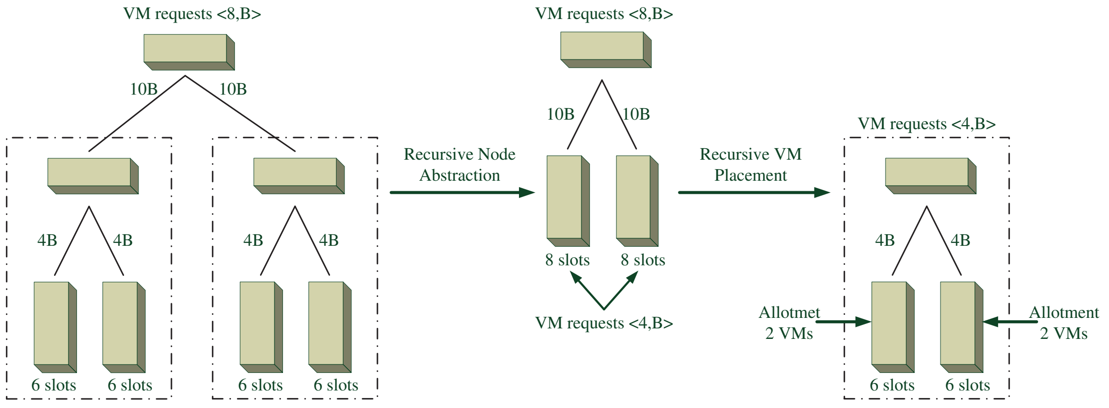


Fig. 5. The process of recursive node abstraction and recursive VM placement

if $L \leq BC$, one PM can support, at most, $\frac{L}{B}$ VMs. Therefore, the number of VMs that one PM could support is:

$$\min\{\frac{L}{B}, C\} \tag{11}$$

Due to the symmetry of the one-layer cluster, this result also applies to both PMs connecting to the same switch. Therefore, adding these two PMs together, for each switch connecting two PMs, we can conclude that the maximum VMs this switch connecting two nodes could support is: $2\min\{\frac{L}{B}, C\}$. We would view this maximal number of VMs as the accumulative capacity of the abstraction node, as shown in Fig. 5.

Based on this abstraction of a bottom-layer switch, we are able to abstract the entire multi-layer cluster to a one-layer cluster. For each switch connecting two sub-trees at each layer from bottom to top, each one-layer cluster can be recursively abstracted into a single node. Upon reaching the root switch at the top, the whole multi-layer cluster is abstracted into a one-layer cluster. In the semi-homogeneous configuration, all the links of the same layer share the same capacity, and all the PMs have the same machine capacity, therefore, for abstraction nodes with the same accumulative capacity, the inner structure of the original sub-trees are the same. We can see in Fig. 5, a cluster consisting of two machines with two lower-layer links are abstracted into a single node of accumulative capacity of 8. Both the abstraction nodes with a capacity of 8 share the same structure of the original abstracted one-layer sub-tree.

B. Hierarchical VM Placement for Multi-layer Clusters

With the abstraction of a K layer multi-layer cluster into a one-layer cluster, we can use the optimal solution for the discussion in Section 3. Based on that result, we propose our hierarchical VM placement algorithm for a multi-layer cluster.

Theorem 1. *For a semi-homogeneous datacenter configuration, if $BC \leq L_K$, the proposed hierarchical VM placement algorithm is optimal to minimize the combinational utilization.*

The proof of optimality is shown in Appendix C. As a matter of fact, the semi-homogeneous configuration with $BC \leq L_K$ is widely adopted in most datacenters.

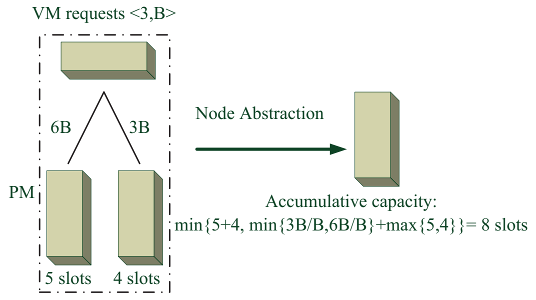


Fig. 6. One-layer heterogeneous cluster

Algorithm 1 Hierarchical VM Placement Algorithm

Input: The links and PMs capacity; VM requests $\langle N, B\rangle$

```
1: for layer $i$=1 to N do
2:     for all switches in layer $i$ do
3:         Calculate the accumulative capacity for each switch
           connecting two sub-trees in the layer
4: if input VMs could be accepted then
5:     for layer $j$=N to 1 do
6:         for all switches in layer $j$ do
7:             Optimally allocate the VMs to this subtree
```

Our algorithm can be divided into two steps. Firstly, for each switch from bottom to top, the accumulative capacity of the abstraction node rooted at that switch is calculated. Upon reaching the top-layer root switch, we obtain an abstracted one-layer cluster. Secondly, for the input, provided that they can be accepted, for each switch connecting two sub-trees at each layer from top to bottom, recursively allocate the input VMs into its two sub-trees according to our optimal result in Eq. 19. Upon finishing the bottom-layer switch (access switch), all the VMs are allocated into the PMs, as shown in Fig. 5. We summarize our algorithm in Algorithm 1.

For each switch at each layer, our algorithm takes a constant time to calculate the optimal solution, according to Eq. 19. For a K layers cluster, layer k has 2^{k-1} switches. Therefore, the total time of calculation is $\sum_{k=1}^{K-1} k$. Suppose we have M machines at the bottom, then $M = 2^k$. Our algorithm takes two loops, one is the abstraction from bottom to top, the other is the placement from top to bottom. One loop takes a time of $O(M)$. Two loops is still $O(M)$. Therefore, the total time complexity of our algorithm is $O(M)$, which is very efficient.

C. Discussion

In a link hungry condition, the abstraction process from bottom to top can be conservative. Therefore, the accumulative capacity of the abstraction node can be smaller than the actual capacity. This will lead to some situations where our algorithm cannot schedule a VM request that should be able to be scheduled. Since our abstraction process is layer-by-layer from bottom to top, each step could be conservative calculated. The boundary situation is that:

Observation 1. *in layer k, if one abstraction node capacity is larger than the sum of layer k's links capacities minus one layer k's link capacity, the accumulative capacity is conservatively calculated for layer k.*

During the abstraction process from the bottom layer to the top, granted that one such situation happened, the whole calculation would be conservative. However, in real datacenters, each PM usually supports 4 VM or 8 VM slots, usually below 10. However, for the link capacity, the bottom layer link is usually 1 Gbps, and the upper layer link capacities are usually more than 10Gbps. On the other hand, one VM usually requests for 100 Mbps. We can see that each link can support more than 10 VMs. The links are usually not hungry; therefore, the conservatively calculation is not a common case.

V. A Heterogeneous Case Study

Now we study the scenario of heterogeneous clusters, where all the capacities of PMs are heterogeneous, along with the link capacities, as shown in Fig. 6. The motivation for studying this scenario is that today's datacenters can support multiple tenants' requests. The VM requests of different tenants may come at different times. After one tenant's VMs are placed into the datacenter, all the links and PMs capacities will change, making the datacenter a heterogeneous configuration, which will make this maximal-elasticity problem an NP-hard problem. However, our hierarchical algorithm is still useful, and could provide a great approximation to the optimal results.

A. One-layer Optimality

Similarly, we firstly study a one-layer cluster under heterogeneous configuration. However, the calculation of accumulative capacity is different, as shown in Fig. 6. With an input $\langle N, B\rangle$, considering the total machine resources of the cluster, we have: $N \leq C_1 + C_2$. Secondly, with the link capacity constraint, we have $N \leq \max\{C_1, C_2\} + \min\{\frac{L_1}{B}, \frac{L_2}{B}\}$. Considering both link and machine capacity limitations, we have:

$$N \leq \min\{\max\{C_1, C_2\} + \min\{\frac{L_1}{B}, \frac{L_2}{B}\}, C_1 + C_2\} \quad (12)$$

This is the maximal number of VMs that this heterogeneous one-layer cluster can support. We still use N^* as the number of VMs that can be accepted into this cluster, as follows:

$$N^* = \min\{N, \max\{C_1, C_2\} + \min\{L_1, L_2\}, C_1 + C_2\} \quad (13)$$

We try to obtain an optimal result for the one-layer cluster. With N^* accepted into this cluster, we allocate x VMs on the left node, leaving $N^* - x$ on the right node. In a heterogeneous scenario, the configuration of a binary cluster is asymmetric, hence, we need to consider all N^*+1 different ways to allocate the VMs. We will discuss the problem separately in the interval of $x \leq \lfloor \frac{N^*}{2} \rfloor$ and $x \geq \lfloor \frac{N^*}{2} \rfloor$, and then lead to the final result.

For $x \leq \lfloor \frac{N^*}{2} \rfloor$, the link utilization is $\max\{x\frac{B}{L_1}, x\frac{B}{L_2}\}$, and the machine utilization is $\max\{x\frac{R}{C_1}, (N^* - x)\frac{R}{C_2}\}$. Then, the combinational utilization of the cluster is:

$$U(x) = \max\{x\frac{B}{L_1}, x\frac{B}{L_2}, x\frac{R}{C_1}, (N^* - x)\frac{R}{C_2}\} \quad (14)$$

The deduction for Eq. 14 is in Appendix B.

B. Binary Abstraction

In the heterogeneous multi-layer clusters, each switch connecting two PMs is asymmetrical. Similar to the result in Eq. 11, one PM with capacity C_1 can support $\min\{C_1, \frac{L_1}{B}\}$ VMs, while the other PM with capacity C_2 can support $\min\{C_2, \frac{L_2}{B}\}$ VM slots. Hence, each switch connecting two PMs can support at most $\min\{C_1, \frac{L_1}{B}\} + \min\{C_2, \frac{L_2}{B}\}$ VMs. We would view this maximal number of VMs as the accumulative capacity of the abstraction node. After we recursively do the abstraction process from bottom to top, we can use the optimal result in Eq. 25 and Algorithm 1 to recursively place the input VMs into each switch. Upon finishing the bottom-layer switch, all VMs are placed in the PMs.

VI. Evaluation

In this section, we conduct two simulations for both semi-homogeneous and heterogeneous datacenter configurations. The topology we use is a three-layer binary tree structure, as in Fig. 3. Our VM placement algorithms are compared with the optimal solution produced by brute-force search.

We conduct one evaluation for the semi-homogeneous configuration. Since we have proven the optimality of our algorithm for the machine resource hungry scenario, we only evaluate under the link hungry scenario. For the link hungry semi-homogeneous scenario, we vary the capacities of the bottom-layer links as 2 Gbps, 4 Gbps, and 6 Gbps. All the links between switches are identical: 10 Gbps. Each VM's bandwidth demand is 1 Gbps. Each PM has 10 VM slots. For the heterogeneous configuration, we vary link capacity. Each link capacity range is [5,10] Gbps, [5,15] Gbps, [5,20] Gbps. Each VM's bandwidth demand is still 1Gpbs.

From Figs. 7, 8, we can see that when the number of VMs increase, the utilization increases. This is because more VMs will consume more resources, leading to the increase in the combinational utilization of the clusters, which will lower the elasticity of VMs. From Fig. 7, when the bottom-layer link capacity increase, the situation of a link-hungry cluster is alleviated. Then, the datacenter can support more VMs, as shown in the comparison in the sub-figures. Besides, we can observe that our algorithm is very close to the optimal value. From Fig. 8, under the heterogeneous scenario, we can see that, as the links and PMs capacities increase, more VMs can be supported by the datacenter. We can see that our algorithm is very close to the optimal solution. In sum, under both link hungry semi-homogeneous and heterogeneous scenarios, our algorithm has a high approximation to the optimal solution, which is likely to lead to a good performance for a multi-tenant datacenter.

VII. Extension: linear elasticity in a K-ary tree

During the discussion above, we focus on a binary tree structure, where each access switch connects with only two PMs. In fact, today's datacenter topology is usually a K-ary tree with K PMs connecting to each access switch at the bottom. Thus, we study the K-ary topology datacenter in a semi-homogeneous configuration.

Besides, in Section II, for the same VM, we allow the bandwidth and machine resource to have the same growth ratio, thereby, the same elasticity. In reality, we might need them to have different growth ratios. Hence, we integrate a linear coefficient c to show the elasticity relationship between bandwidth and machine resource. In that case, the combinational elasticity of VM_i is $E_i = \min\{E_i^r, cE_i^l\}$.

A. K-ary One-layer cluster optimality

Firstly, we focus on a one-layer K-ary tree to obtain the one-step optimal result. Suppose that, we have N input VMs to place into the one-layer K-ary cluster. Similar to the analysis for the binary-tree, the maximal number of VMs that can be accepted into the cluster is: $max\{K\min\{\frac{L}{B}, C\}, \frac{L}{B}+C\}$.

Given N input VMs, let N^* denote the actual number of VMs accepted into this K-ary cluster. Assume that the i^{th} PM is allocated to x_i VMs. Due to the symmetry of this K-ary tree, without of loss of generality, we assume that $x_1 < x_2 <, ..., < x_K$. Therefore, for the machine utilization, Eq. 6 can be transferred as $\frac{x_K}{C}$. Also, for the bandwidth utilization, Eq. 7 could be transferred as: $\min\{x_K, N-x_K\} * \frac{B}{L}$. Then, considering the linear elasticity relationship between bandwidth and machine resource, the combinational utilization for PMs and links in Eq. 8 is:

$$\max\{\frac{x_K}{C}, \min\{x_K, N-x_K\} * \frac{B}{cL}\} \tag{15}$$

Since we focus on worst-case, therefore, we minimize the worst-case combinational utilization in Eq. 15. In that case, we only pay attention to the allotment of x_K. Hence, Similar to the binary-tree analysis, we can obtain the optimal result to minimize Eq. 15, as shown in Appendix D.

B. K-ary abstraction

After obtaining the one-step optimal result, we can use the bottom-to-top abstraction and the top-to-bottom placement to allocate the input VMs into the PMs. Similar to the result in Eq. 11, each node can support $\min\{C, \frac{B}{L}\}$ VMs. Hence, each switch connecting K nodes can support at most $K\min\{C, \frac{B}{L}\}$ VMs. We can also abstract this multi-layer K-ary cluster into a one-layer K-ary cluster. For each switch from the bottom layer to the top, we recursively abstract it along with its K sub-trees into one node. After that, we use the optimal result in Eq. 34 and Algorithm 1 to recursively place the input VMs into each switch. Upon finishing the bottom-layer switch, all VMs are placed in the PMs.

VIII. Related Work

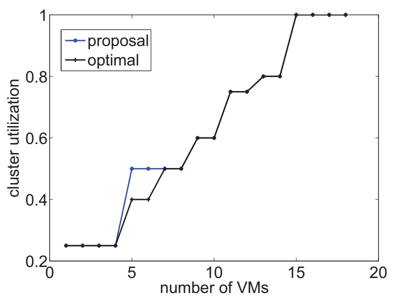
Much work has been done regarding VM placement in the cloud computing environment, which is a complicated task involving various constraints, including performance [4], availability [5], and so forth. Among all these constraints, network is one important concern in the VM placement problem. In [6], the author proposes minimizing the traffic cost through VM placement. Their objective is to place VMs that have large communication requirements close to each other, so as to reduce network capacity needs in the datacenter. Oktopus [3] uses the hose model to abstract the tenant's bandwidth request, including both virtual cluster and oversubscribed virtual clusters. They propose a VM allocation algorithm to deal with homogeneous bandwidth demands, which is also what we are using in this paper. The virtual cluster provides tenants with guarantees on the network bandwidth they demand, which, according to [7], could be interpreted as the min-guarantee requirements. However, this min-guarantee fails to consider the potential growth of a tenant's network demand. In that case, the virtual cluster allocation based on this min-guarantee policy will not have enough resources to accommodate tenants' future growth demands, which can lead to the loss of customers. In order to alleviate this problem, we propose the concept of elasticity, which considers existing customers' growing bandwidth demand in the future.

IX. Conclusion

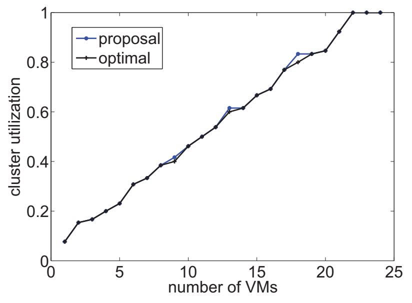
In this paper, we study the resource management problem on the cloud datacenter, which is now suffering from both limited resources and the variety of users' requests. Compared to the previous work, we focus on guaranteeing the on-demand scaling of cloud computing, and we propose the concept of elasticity of the VM requests. To maximize the elasticity of the input VMs, we transfer it into the utilization minimization problem, and propose our hierarchical VM placement algorithm. We study the scheduability of the input VMs and also prove the optimality of our algorithm under a frequently used datacenter configuration. Furthermore, we also conduct a study on the heterogeneous scenario to meet the requirements of a multi-tenant datacenter. The evaluation results show the high efficiency of our algorithm. At last, we extend our algorithm into the K-ary datacenter topology.

Acknowledgement

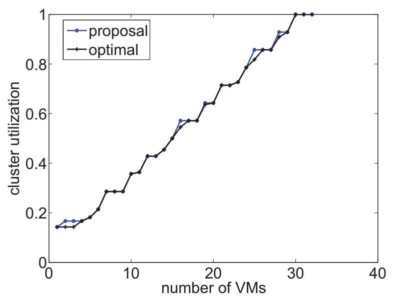
This research was supported in part by NSF grants ECCS 1231461, ECCS 1128209, CNS 1138963, CNS 1065444, and CCF 1028167.



(a) bottom-layer links capacities: 2 Gbps

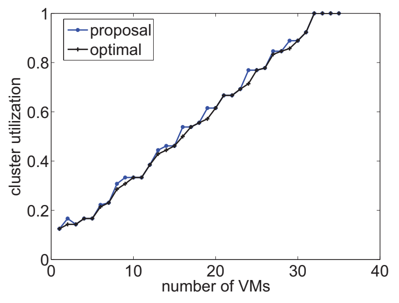


(b) bottom-layer links capacities: 4 Gbps

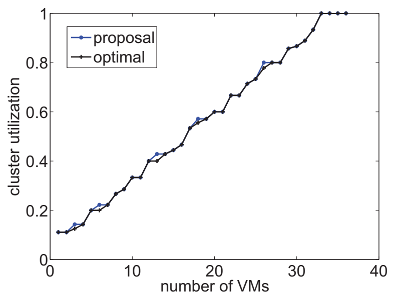


(c) bottom-layer links capacities: 6 Gbps

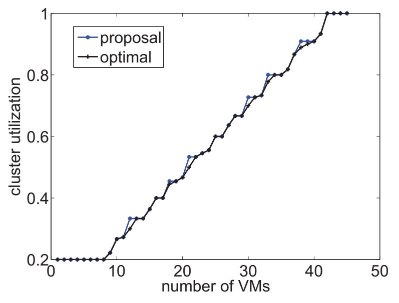
Fig. 7. Performance evaluation: semi-homogeneous configuration



(a) link capacity range: [5,10] Gbps



(b) link capacity range: [5,15] Gbps



(c) link capacity range: [5,20] Gbps

Fig. 8. Performance evaluation: heterogeneous configuration

REFERENCES

[1] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, (New York, NY, USA), pp. 202–208, ACM, 2009.

[2] J. Zhu, D. Li, J. Wu, H. Liu, Y. Zhang, and J. Zhang, "Towards bandwidth guarantee in multi-tenancy cloud computing networks," in *Proc. of ICNP 2012*, pp. 1 –10.

[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. of the ACM SIGCOMM 2011*, pp. 242–253.

[4] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Proc. of ICAC 2008*, pp. 3 –12.

[5] E. Bin, O. Biran, O. Boni, E. Hadad, E. Kolodner, Y. Moatti, and D. Lorenz, "Guaranteeing high availability goals for virtual machine placement," in *Proc. of ICDCS 2011*, pp. 700 –709.

[6] 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*, pp. 1–9, 2010.

[7] L. Popa, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: sharing the network in cloud computing," in *Proc. of ACM HotNets-X 2011*, pp. 22:1–22:6.

APPENDIX A
ONE-LAYER HOMOGENEOUS CLUSTER OPTIMAL RESULT

The objective function is:

$$Minimze\ U(x) = \max\ \{x\frac{B}{L}, \frac{N-x}{C}\} \tag{16}$$

With link and PM capacity limits, the domain of x is:

$$x \in [\max\{N^* - C, 0\}, \min\{\frac{L}{B}, \lfloor\frac{N^*}{2}\rfloor\}] \tag{17}$$

The minimal point for $U(x)$ can be obtained, when x is:

$$x^* = \frac{\frac{L}{B}}{\min\{C, \frac{L}{B}\} + C}\ N^* \tag{18}$$

Apparently, x^* is within the domain, which is the optimal solution to allocate the N^* into two machines. However, x^* in Eq. 18 may not be an integer. In that case, we compare the value of $U(x)$ under both $x = \lfloor x^* \rfloor$ and $x = \lceil x^* \rceil$. So the optimal solution for maximizing the elasticity is:

$$x^* = \begin{cases} \lfloor x^* \rfloor, U(\lfloor x^* \rfloor) \le U(\lceil x^* \rceil) \\ \lceil x^* \rceil, U(\lfloor x^* \rfloor) \ge U(\lceil x^* \rceil) \end{cases} \tag{19}$$

APPENDIX B
ONE-LAYER HETEROGENEOUS CLUSTER OPTIMAL RESULT

The objective function is:

$$Minimze\ U(x) = \max\{x\frac{B}{L_1}, x\frac{B}{L_2}, x\frac{R}{C_1}, (N^* - x)\frac{R}{C_2}\} \tag{20}$$

The domain of x is:

$$x \in [\max\{\min\{N^* - C_2, C_1\}, 0\}, \min\{\frac{L_1}{B}, \frac{L_2}{B}, \frac{N^*}{2}\}] \tag{21}$$

There is a minimal point for $U(x)$, when x is:

$$x_l^* = \frac{\min\{\frac{L_1}{B}, \frac{L_2}{B}, C_1\}}{\min\{\frac{L_1}{B}, \frac{L_2}{B}, C_1\} + \min\{\frac{L_1}{B}, \frac{L_2}{B}, C_1, C_2\}} N^* \tag{22}$$

Obviously, x_l^* is within the domain of Eq. 21. Still, the value of x^* from Eq. 22 may not be an integer. We choose both values of $\lfloor x_l^* \rfloor$ and $\lceil x_l^* \rceil$ for future comparison with another interval. Similarly, for $\lfloor \frac{N^*}{2} \rfloor \le x \le N^*$, the objective function for the utilization of the cluster is:

$$U(x) = \max\{(N^* - x)\frac{B}{L_1}, (N^* - x)\frac{B}{L_2}, (N^* - x)\frac{R}{C_1}, x\frac{R}{C_2}\}$$

The domain of x is:

$$x \in [\max\{\min\{N^* - C_2, C_1\}, \frac{N^*}{2}\}, \min\{\frac{L_1}{B}, \frac{L_2}{B}, N\}] \quad (23)$$

To minimize $U(x)$, we have:

$$x_r^* = \frac{C_2}{\min\{\frac{\min\{L_1, L_2, C_1\}}{B}, C_2\} + C_2} N^* \quad (24)$$

We will choose the values of $\lfloor x_l^* \rfloor$, $\lceil x_l^* \rceil$,$\lfloor x_r^* \rfloor$ and $\lceil x_r^* \rceil$ for comparison. Therefore, the optimal result $x^*$of allocation for a one-layer heterogeneous cluster is shown below:

$$x^* = \begin{cases} \lfloor x_l^* \rfloor, U_l^h(\lfloor x_l^* \rfloor) = \min\{U_l^h(\lfloor x_l^* \rfloor), U_l^h(\lceil x_l^* \rceil), \\ \qquad U_r^h(\lfloor x_r^* \rfloor), U_r^h(\lceil x_r^* \rceil)\} \\ \lceil x_l^* \rceil, U_l^h(\lceil x_l^* \rceil) = \min\{U_l^h(\lfloor x_l^* \rfloor), U_l^h(\lceil x_l^* \rceil), \\ \qquad U_r^h(\lfloor x_r^* \rfloor), U_r^h(\lceil x_r^* \rceil)\} \\ \lfloor x_r^* \rfloor, U_r^h(\lfloor x_r^* \rfloor) = \min\{U_l^h(\lfloor x_l^* \rfloor), U_l^h(\lceil x_l^* \rceil), \\ \qquad U_r^h(\lfloor x_r^* \rfloor), U_r^h(\lceil x_r^* \rceil)\} \\ \lceil x_r^* \rceil, U_r^h(\lceil x_r^* \rceil) = \min\{U_l^h(\lfloor x_l^* \rfloor), U_l^h(\lceil x_l^* \rceil), \\ \qquad U_r^h(\lfloor x_r^* \rfloor), U_r^h(\lceil x_r^* \rceil)\} \end{cases} \quad (25)$$

Appendix C
Theorem 1 Proof

A. A two-layer optimality

Suppose that the capacity of each machine is C in a two-layer cluster, the bandwidth capacity is B_1 for the upper layer links, and B_0 for the lower layer. We have $B_1 \ge B_0$. Here we adopt the unified measuring unit of VM slot. Given N VM requests, without loss of generality, we assume each VM requires machine resource of one unit of C, and bandwidth of one unit of B_1 or B_0. Also, we assume that N VMs requests are schedulable. Here, we study the case where $B_0 \ge C$.

Suppose that, based on our algorithm, the allotment for the four machines is $[a, b, c, d]$. With the symmetric characteristic of binary-tree, according to our algorithm, we always allocate the larger allotment of the given input on the right side without loss of generality. Then, we have $a \le b \le c \le d$.

Assume our algorithm is not optimal, therefore, there is another allocation for the four machines of $[a', b', c', d']$, that is better than our algorithm in minimizing the combinational utilization in Eq. 10. We assume $a' \le b' \le c' \le d'$. Due to the symmetry of binary-tree, we could swap each PM's allotment, so as to make $[a', b', c', d']$ corresponds to $[a, b, c, d]$.

Based on our algorithm, for the first allocation step, we allocate the N requests into the two abstraction nodes of representing the two sub-trees of the root switch. By Eq. 11, the accumulative capacity of each abstraction node is $min\{2B_0, 2C\} = 2C$. For the one-layer cluster, our algorithm optimally allocates the N VMs based on the results of Eq. 19. Since $a \le b \le c \le d$ and $a' \le b' \le c' \le d'$, therefore, for the first step, we have:

$$\max\ \{\frac{a+b}{B_1}, \frac{c+d}{2C}\} \le \max\ \{\frac{a'+b'}{B_1}, \frac{c'+d'}{2C}\} \quad (26)$$

Then, After finishing the allocation for the top layer, according to our algorithm, we are about to do the allocation for the two switches in the second layer. The combinational utilization for the second-layer switch is $\max\ \{\frac{a+b}{B_1}, \frac{d}{C}, \frac{\min\{a+b+c,d\}}{B_0}\}$. Assuming our algorithm is not optimal, we have:

$$\begin{aligned} &\max\ \{\frac{a+b}{B_1}, \frac{d}{C}, \frac{\min\{a+b+c,d\}}{B_0}\} \\ &\ge \max\ \{\frac{a'+b'}{B_1}, \frac{d'}{C}, \frac{\min\{a'+b'+c',d'\}}{B_0}\} \end{aligned} \quad (27)$$

Since $B_0 \ge C$, the result in Eq. 19 will evenly divide the input of the second-layer switch. Therefore, we have $a=b$, $c=d$. Then, $\min\{a+b+c, d\}=d$, and $\frac{d}{C} \ge \frac{d}{B_0}$. Therefore, we have:

$$\max\{\frac{a+b}{B_1}, \frac{d}{C}, \frac{\min\{a+b+c,d\}}{B_0}\} = \max\{\frac{a+b}{B_1}, \frac{d}{C}\} \quad (28)$$

Since $d=c$, then, $\frac{d}{C} = \frac{c+d}{2C}$. Combing Eq. 28, we have:

$$\max\ \{\frac{a+b}{B_1}, \frac{d}{C}, \frac{\min\{a+b+c,d\}}{B_0}\} = \max\ \{\frac{a+b}{B_0}, \frac{c+d}{2C}\} \quad (29)$$

Since $a' \le b' \le c' \le d'$, therefore, $d' \ge c'$. Then, $\frac{d'}{C} \ge \frac{c'+d'}{2C}$. Therefore, we have:

$$\begin{aligned} &\max\{\frac{a'+b'}{B_1}, \frac{d'}{C}, \frac{\min\{a'+b'+c',d'\}}{B_0}\} \\ &\ge \max\{\frac{a'+b'}{B_1}, \frac{c'+d'}{2C}\} \end{aligned} \quad (30)$$

Hence, combing Eqs. 29, 30

$$\begin{aligned} &\max\ \{\frac{a'+b'}{B_1}, \frac{d'}{C}, \frac{\min\{a'+b'+c',d'\}}{B_0}\} \\ &\ge \max\ \{\frac{a'+b'}{B_1}, \frac{c'+d'}{2C}\} \ge \max\ \{\frac{a+b}{B_1}, \frac{c+d}{2C}\} \\ &= \max\ \{\frac{a+b}{B_1}, \frac{d}{C}, \frac{\min\{a+b+c,d\}}{B_0}\} \end{aligned} \quad (31)$$

A contradiction with the assumption in Eq. 27, therefore, the proof is complete for $B_0 \ge C$.

B. Generalization of Optimality

From one-layer to two-layer, the potential factor that can change the optimality of this greedy step is the change of resource capacity, since we use the accumulative capacity to complete the first step. In other words, $\frac{c+d}{C_M} = \frac{c+d}{2C}$ can be smaller than $\max\{\frac{\min\{a+b+c,d\}}{B_0}, \frac{d}{C}\}$. However, according to the allocation scheme of our algorithm, we have proven that $\max\{\frac{\min\{a+b+c,d\}}{B_0}, \frac{d}{C}\}$ is equal to $\frac{c+d}{C_M}$, which guarantees the generalization of optimality for one-step. Since all PMs' capacities are the same, and the capacity of the links at the same layer are the same, from the view of physical meaning, our abstraction process misses no information of the lower layer resources for both links and machines. Hence, we conclude that the optimality is secured for one-step generalization. Based on this observation, we can use the induction method to prove the optimality of our algorithm in a k layer cluster. When $k = 1$, it is a one-layer cluster, and the optimal solution is given. Suppose that, for the k^{th} layer, our algorithm is optimal, then, the optimality from the k^{th} to $k^{th} + 1$ layer is also secured. Therefore, our algorithm is optimal.

Appendix D
K-ary One-layer cluster optimal result

The objective function is:

$$Minimze\ U(x_K) = \max\{\frac{x_K}{C}, \min\{x_K, N - x_K\} * \frac{B}{cL}\} \quad (32)$$

With link and PM capacity limits, the domain of x_K is:

$$x_K \in [\frac{N^*}{K}, \min\{\frac{L}{B}, C\}] \quad (33)$$

The minimal $U(x_K)$ can be obtained, when x_K is:

$$x_K^* = \begin{cases} \frac{1}{K}N^*, & C \geq c\frac{L}{B} \\ \frac{C}{c\frac{L}{B}+C}N^*, & C \leq c\frac{L}{B} \end{cases} \quad (34)$$

Obviously, x_K^* is within the domain, which is the optimal solution to allocate the worst-case machine. Of course, there is also an integer problem, as before. We could use a similar method to round the result into the integer. Due to the space limitation, we do not repeat it here.