On Maximum Elastic Scheduling of Virtual Machines for Cloud-based Data Center Networks

Jie Wu†, Shuaibing Lu*†, and Huanyang Zheng†
†Center for Networked Computing, Temple University, USA
*College of Computer Science and Technology, Jilin University, China

***Abstract*—Task resource allocation has always been an important issue in cloud-based data center networks (DCNs). This paper considers provisioning the *maximum admissible load* (MAL) of virtual machines (VMs) in physical machines (PMs) with underlying tree-structured DCNs using the hose model for communication. The limitation of static load distribution is that it assigns tasks to nodes in a once-and-for-all manner, and thus, requires a priori knowledge of program behavior. To avoid load redistribution during a run time where the load grows, we introduce *maximum elasticity scheduling*, which has the maximum growth potential subject to the node and link capacities. Given a tree-based topology, this paper aims to find the schedule with the maximum elasticity across both nodes and links. A distributed linear solution has been found, and we discuss several extensions of the model. We conclude the paper by presenting various simulation results.**

***Index Terms*—Data center networks (DCNs), cloud, distributed algorithms, elasticity, hose model, optimization.**

I. Introduction

Task resource allocation has always been an important issue in cloud-based data center networks (DCNs). Virtual machines (VMs) scheduling is one popular model that optimizes a chosen metric subject to the resource limitations of both physical machines (PMs) and links [1]. This paper enhances a new QoS metric called *maximum elastic scheduling*, a task-assignment scheme that supports maximum uniform growth in both computation and communication without resorting to task reassignment. This model was originally proposed in [2], but its optimal solution is limited to a semi-homogeneous tree structure. In this paper, we enhance the optimal solution to a general heterogeneous case.

We model the network as a tree T in a typical DCN. Each leaf node is a physical machine and each internal node is a switch. A load at a leaf node is called a *computation load* and determines the communication load. We use the *hose model* [3] for communication where each node has an aggregated performance guarantees to the set of all other nodes. Figure 1 (b) shows a two-level, three-node binary tree where each PM (a leaf node) is represented by a slotted rectangle (e.g., VM slots or computation loads) and each internal node (switch) is represented as a circle. Numbers associated with nodes and links are available VM slots and communication bandwidth, respectively. Each VM has B Gbps total communication with other VMs. The communication loads of the left link and the right link are the same: it is the lesser of the assigned VMs in the two leaf nodes multiplying

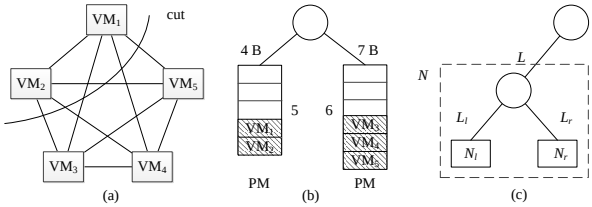

Fig. 1. (a) VM scheduling as a cut, (b) virtual node, and (c) aggregation tree.

B based on the hose model. This is analogous to the maximum flow in a cut as shown in Figure 1 (a). If the left leaf node of Figure 1 (b) is assigned 2 VMs of one part of the cut and the right leaf node is assigned 3 VMs of the other part, the communication load is $min\{2B, 3B\} = 2B$. If the left leaf node is assigned 1 VM and the right 4 VM, the communication load is $min\{1B, 4B\} = B$. When the left and right node loads are doubled in Figure 1 (b), the communication load becomes $4B$. This load can still be handled by the left link and the right link. However, if we allocate the computation load based on the node capacity, the corresponding communication load is $min\{5B, 6B\} = 5B$, which exceeds $4B$ the available bandwidth of the left link.

Using the hose model, we study two provisioning problems:

1) Given a graph T with available node and link capacities, what is the *maximum admissible load* (MAL) of T under the hose model?
2) Given a load that is admissible, what is the optimal schedule so that the uniform growth rate at all leaf nodes is maximized under the capacity constraint?

The optimal schedule for the second problem is called the schedule with the *maximum elasticity*. The MAL in Figure 1 (b) is 10, with 4 VMs (also loads) assigned to the left leaf node and 6 loads to the right leaf node. Both the left link and the right leaf node reach maximum capacities. Suppose that we now have a load of 5 to be assigned, which is below the MAL. The schedule with maximum elasticity assigns 2 loads to the left leaf node and 3 loads to the right leaf node, as shown in Figure 1 (b). The maximum elasticity is 100 percent. That is, each side can be doubled without violating the node or link capacities. We introduce the concept of an *aggregation tree*, which is used to calculate the maximum link bandwidth needed for each link under the hose model (given that the workload at each leaf node is known a priori). This aggregation tree gives us a simple iterative solution that abstracts each two-level, three-node branch (such as the one in Figure 1 (b)) into one *virtual node*. The abstraction is a

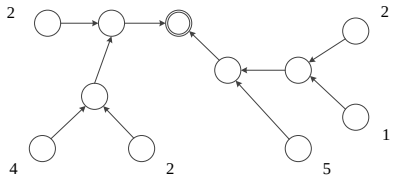

Fig. 2. The aggregation tree based on the hose model.

bottom-up aggregation process that determines the MAL at the root of the tree; the schedule with the maximum elasticity is decided in a top-down partition process starting at the root. We later refine this process since the orientation of the aggregation may not coincide with the traditional orientation of a full binary tree. We propose a distributed optimal solution that uses only three copies of the simple solution to compute different orientations of a tree with different roots. Both computation and communication complexity are linear to n where n is the number of leaf nodes in the tree.

To simplify the discussion, we assume $B = 1$ for each VM. Although the schedule with the maximum elasticity in DCNs can be solved by the classic linear programming (LP), we strive to find a simple and efficient solution, similar to the Bellman-Ford solution to the shortest path problem.

II. A Simple Solution

We suppose that the workload of each node is given for a binary tree (or simply tree). The hose-model-based orientation of each link is determined as follows: if the link is used as the cut, the graph can be partitioned into two parts; the link orientation is the end node that has no more than 50% of the workload, and the other end node has no less than 50% of the workload. In case of a tie (where each end node has exactly 50% of the workload), the node with the smaller ID points towards the one with a larger ID. Note that under the hose model $B = 1$, the communication load is determined through a *cut* on the link: it is the lesser part of two computation loads (one from each side of the cut). In Figure 2, the root is represented as a double cycle (suppose that the link bandwidth is infinity). The workload from the left branch (pointing towards the root) is $2 + 4 + 2 = 8$, and the right branch has the same workload.

Theorem 1. *The hose-model-based link orientation of a tree is defined as a directed tree, and each leaf node in this directed tree has a directed path to a single root.*

Proof. If a leaf node has no more than 50% of the total workload, it points towards its neighbor. In addition, its workload will be added to its neighbor and then adds the workload to the neighbor. (This also occurs in cases where the leaf node has exactly 50% of the workload and has a smaller ID.) If the leaf node has no less than 50% of the total workload, it points toward itself. This aggregation process eventually stops at a node (called root) that has more than 50% of the aggregated workload . (In cases where the leaf node has exactly 50% of the workload, this occurs when the ID is larger). It is clear that two roots are impossible due to the majority workload requirement. □

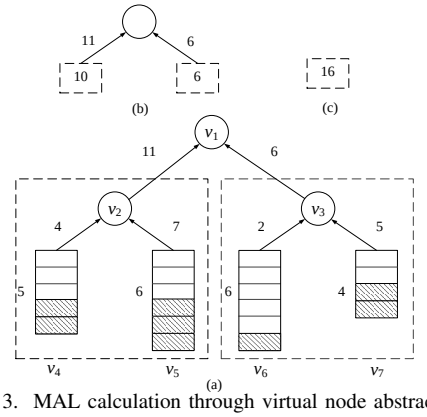

Fig. 3. MAL calculation through virtual node abstraction.

The above proof shows a unique aggregation process that calculates the maximum bandwidth needed at each link for a given set of workloads. It also serves as the basis of our iterative calculation process for MAL when the root is given. The directed tree that is used to represent the orientation is called an aggregation tree. The simple solution iteratively abstracts the given tree in a bottom-up manner. As shown in Figure 1 (c), the basic unit of the abstraction is a two-level, three-node branch that becomes one virtual node at the higher level. In this abstraction, one internal node and two virtual nodes serve as the child nodes of the internal node. At the bottom level of the tree, a virtual node is a leaf node. At all other levels, a virtual node is abstracted from the branch rooted at the same node. Suppose N_l (L_l) and N_r (L_r) have available node space (link bandwidth) for the left and right virtual nodes, respectively, as shown in Figure 1 (c).

$$N = min\{N_l, L_l\} + min\{N_r, L_r\} \tag{1}$$

The minimization operation ensures that the value of each branch satisfies both node space and link bandwidth requirements. This abstraction process continues level-by-level until G is reduced to a single virtual node. The available node capacity of this virtual node is the MAL. Figure 3 shows this process on a three-level full binary tree. The load of the left virtual node in Figure 3 (a) is determined through the left-subtree: $10 = min\{5, 4\} + min\{6, 7\}$. Similarly, the final load of the virtual node shown in Figure 3 (c) is decided by abstraction from Figure 3 (b). In this case, the MAL is $16 = min\{10, 11\} + min\{6, 6\}$.

Once the MAL is determined, we can iteratively determine the actual schedule that achieves the maximum elasticity. The process is now top-down. An example for a complete binary tree is shown in Figure 3; supposing that N is a given load that is no more than the MAL, we partition the load into two parts. The left and right subtrees are assigned $min\{N_l, L_l\}/N$ and $min\{N_r, L_r\}/N$ portions of the total load, respectively. This process continues for each of the subtrees until the given load is eventually assigned to all leaf nodes. In Figure 3, let 8 be a given load that is less than the MAL 16. Based on Figure 3 (b), the load ratio of the left subtree to the right subtree should be $10 : 6$. This means that the left subtree is assigned a load of 5 and the right subtree is assigned a load of 3. This process

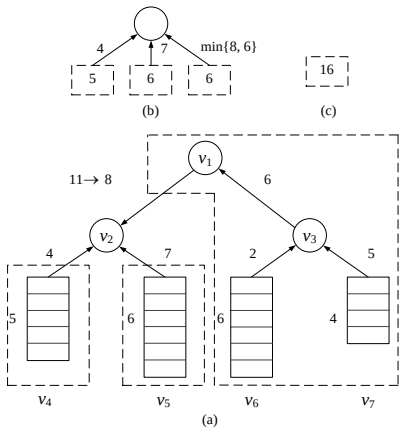

Fig. 4. Optimal solution through a different orientation of aggregation.

continues at the next level. The final assignment is in Figure 3 (a), and the load assignment is shown as shaded slots. This schedule corresponds to the one with the maximum elasticity, which is 100 percent of the current load. The load assignments to v_6 and v_7 are 1 and 2, respectively. This is not proportional to their available node spaces.

III. An Optimal Solution

The simple solution gives us some key ideas about iterative abstractions. The complexity is $O(n)$, where n is the number of leaf nodes. In fact, each internal node (out of $n-1$) needs to calculate twice, performing one bottom-up aggregation for the maximum admissible load and one top-down load distribution. However, the simple solution may not generate the schedule with the maximum elasticity. Figure 4 shows a slightly revised version of the example in Figure 3, changing the upper left link capacity from 11 to 8. In this case, the MAL reduces to $8+6 = 14$ based on the simple solution. As shown in Figure 4, if we use node v_2 as the "root" instead of node v_1 (the binary tree becomes a ternary tree), the new MAL is 16 when using the same approach.

To obtain the optimal solution, we can apply the simple solution to different orientations of the aggregation tree and select the best orientation (i.e., the one with the maximum MAL). An orientation is determined by selecting the root of the tree. The orientation of each link points towards the selected root for aggregation, as shown in Figure 4 (v_2 is the root). Since there are $n-1$ internal nodes and n leaf nodes that can be roots, applying the simple solution directly $2n-1$ times is not efficient. Here, we introduce a distributed solution that applies the simple solution three times to obtain the optimal solution. The key idea is that a node with three branches (e.g., the node v_2 in Figure 4) has, at most, three orientations. When a three-branch node receives virtual load information from two branches, it passes the information on to the connected node in the remaining branch. When a two-branch node (only one such node exists in a *strict* binary tree; a binary tree is considered strict if all the nodes but the leaf nodes have two children) receives virtual load information from one branch, it passes the information on to the connected node in the remaining branch. The MAL at an internal node is the summation of the loads from all branches. The MAL at a leaf node is the summation of its branch and its local computation load.

Optimal solution

(At a leaf node with node capacity N)

- **Send** its load **to** the connected internal node.
- /* Upon receiving a virtual load N_1 from the only branch with available link bandwidth L_1 */
  Calculate its MAL: $min\{N, \infty\} + min\{N_1, L_1\}$

(At the internal node with two branches)

- /* Upon receiving a virtual load N_i from a branch with available link bandwidth L_i, $i : 1, 2$ */
  Send virtual load $min\{N_i, L_i\}$ **to** the other branch.
- /* Upon receiving virtual load, N_1 and N_2, from two branches */
  Calculate its MAL: $min\{N_1, L_1\} + min\{N_2, L_2\}$.

(At an internal node with three branches)

- /* Upon receiving virtual load, N_i and N_j, from two branches, with $i, j(\neq i) : 1, 2, 3$ */
- **Send** $min\{N_i, L_i\} + min\{N_j, L_j\}$ **to** the third branch
- /* Upon receiving virtual load, N_1, N_2, and N_3, from all branches */
  Calculate its MAL: $min\{N_1, L_1\} + min\{N_2, L_2\} + min\{N_3, L_3\}$.

In the optimal solution, the root corresponds to a three-branch aggregation. To verify that each link can handle the computation load, we consider a cut to one of the three adjacent links, say on the branch with index 1. Based on the hose model, the computational load on branch 1 is the cut value: $min\{min\{N_1, L_1\}, min\{N_2, L_2\}+min\{N_3, L_3\}\}$ (this is no more than L_1). In Figure 4 (b), the communication load at the middle branch is bounded by $min\{min\{6, 7\}, min\{5, 4\} + min\{6, 6\}\} = 6$, which is no more than its available communication bandwidth.

A root selection example for a two-level, three-node tree is illustrated in Figure 1 (c). The MAL at the center node is $min\{N_l, L_l\} + min\{N_r, L_r\}$. The MAL at the left leaf node is $min\{N_l, \infty\} + min\{min\{N_r, L_r\}, L_l\}$. The MAL at the right leaf node is $min\{N_r, \infty\}+min\{min\{N_l, L_l\}, L_r\}$. The internal load of a leaf node can be considered a branch with infinite bandwidth. When N_l, N_r, L_l, and L_r are 8, 4, 5, and 4, the MAL at the left leaf node, center node, and right leaf node are 12, 9, and 8, respectively. In this case, the left leaf node is the root and has a maximum MAL of 12. When the load is 6, a load, $6 \times 8/12 = 4$ is assigned to the left leaf node. The remaining load, $6 \times 4/12 = 2$, is assigned to the right leaf node. The maximum growth rate is 100 percent.

In the following optimal solution, each leaf node initiates calculation by sending its available load to the connected internal node. Table I shows an example of a step-by-step calculation of the MAL for each node in Figure 4. The final MAL is the maximum MAL among the MALs calculated at all internal nodes. Once the MAL is determined with the selected root node, the schedule with the maximum elasticity

TABLE I
STEP-BY-STEP CALCULATION OF MALS FOR THE EXAMPLE OF FIGURE 4

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
Step 1	-	-	-	**send** 5 **to** v_2	**send** 6 **to** v_2	**send** 6 **to** v_3	**send** 4 **to** v_3
Step 2	-	**send** $\min\{5,4\}$+ $\min\{6,7\}$=10 **to** v_1	**send** $\min\{6,2\}$+ $\min\{4,5\}$=6 **to** v_1	-	-	-	-
Step 3	**send** $\min\{6,6\}$ =6 **to** v_2 **send** $\min\{10,8\}$ =8 **to** v_3	-	-	-	-	-	-
Step 4		**send** $\min\{6,8\}$+ $\min\{6,7\}$=12 **to** v_4 **send** $\min\{6,8\}$+ $\min\{5,4\}$ =10 **to** v_5	**send** $\min\{8,6\}$+ $\min\{4,5\}$=10 **to** v_6 **send** $\min\{8,6\}$+ $\min\{6,2\}$=8 **to** v_7	-	-	-	-
MAL	$\min\{10,8\}$+ $\min\{6,6\}$=14	$\min\{5,4\}$+$\min\{6,7\}$ + $\min\{8,6\}$=**16**	$\min\{6,2\}$+$\min\{4,5\}$ + $\min\{8,6\}$=12	$\min\{12,4\}$+ $\min\{5,\infty\}$=9	$\min\{10,7\}$+ $\min\{6,\infty\}$=13	$\min\{10,2\}$+ $\min\{6,\infty\}$=8	$\min\{8,5\}$+ $\min\{4,\infty\}$=9

is the same as the schedule in the simple solution. That is, the schedule is based on the proportion of the virtual load at each branch. In the example in Figure 4, we suppose that 8 is the given load. The load distribution is based on the branch capacities (i.e., $\min\{N_i, L_i\}$ for $i \in \{1,2,3\}$) of three branches, as shown in Figure 4 (b). The left, middle, and right branches are assigned $8 \times 4/16 = 2$, $8 \times 6/16 = 3$, and $8 \times 6/16 = 3$, respectively. The right branch further partitions its assigned load to nodes v_6 and v_7 using the same method. Eventually, v_4, v_5, v_6, and v_7 are assigned loads of 2, 3, 1, and 2, respectively. Each of these loads can grow at a maximum rate of $(16-8)/8 \times 100 = 100$ percentage.

IV. PROPERTIES

This section studies several properties of the optimal solution and the simple solution.

Theorem 2. *The optimal solution determines the MAL for a given binary tree under the hose model.*

Proof. Based on the optimal solution and Theorem 1, we only need to show that the MAL determined at each node v is the MAL with v as the root in the corresponding orientation tree. We prove by induction from the bottom level to the top level. Consider each two-level, three-node tree at the bottom two levels of the tree. We first show that $min\{N_l, L_l\}+min\{N_r, L_r\}$ is the maximum virtual load. We prove this by contradiction. Suppose we can add a small δ to this virtual load; at least a portion of δ is added to one branch, say the left branch in this example, without loss of generality. This portion plus $min\{N_l, L_l\}$ overloads either the virtual node (leaf node in this case) or the left link. Suppose the theorem holds for up to the level $k-1$, for $k > 2$. At the level k, we again consider two levels; level k with internal nodes and level $k-1$ with virtual nodes. The argument for the base case still applies to each two-level, three-node tree, and thus, this theorem is proved through induction. □

Theorem 3. *For a given admissible load, the hierarchical load distribution from the root to the leaf nodes, based on the virtual load proportions of each branch, generates a schedule with maximum elasticity.*

Proof. Similar to Theorem 2, we prove by induction. The key difference is that for each two-level, three-node subtree, the load distribution based on the load proportion of the left and right branches is optimal. The fact that the proportions are determined by the maximum load of both the left and right branches proves this. Any deviation from this proportion reduces the growth rate of either the left or the right branch. As a result, the elasticity is reduced in either case. □

Theorem 4. *The optimal solution uses* $2\log n+1$ *steps, where* n *is the number of leaf nodes for a given full-binary tree. The computation complexity is* $5(n-1)$*, and the communication complexity is* $4(n-1)$.

Proof. In the optimal solution, all information propagation is loop-free starting from a leaf node and ending at an internal node or a leaf node. The longest distance is the diameter of the tree. Since each node, internal or leaf, needs to determine its MAL, one extra step is needed. In a full binary tree, the diameter is $2\log n$ where n is the number of leaf nodes. Therefore, the optimal solution uses $2\log n + 1$ steps. Since n is the number of leaf nodes in the tree, the number of internal nodes is $n-1$. Each internal node computes only when it passes virtual load information to another node. Since there are $n-2$ links connecting internal nodes and each link is bi-directional, there are $2(n-2)$ computation steps for virtual load calculation. Each leaf node receives virtual load information from an internal node after computation a total of n times. In addition, each node calculates its MAL once ($2n-1$ in total). Therefore, the total computation cost is $5(n-1)$. Each leaf node communicates once for a total of n. Each link that connects two internal nodes communicates twice (once in each direction) to pass along virtual load information for a total of $2(n-2)$. Each leaf node receives virtual load information once from an internal node for a total of n. Therefore, the total communication cost is $4(n-1)$. □

When we consider elasticity, the bottleneck must be either in the link or the node in terms of capability of growth. Next, we consider two special situations under which the simple solution is optimal. Let us first introduce two special

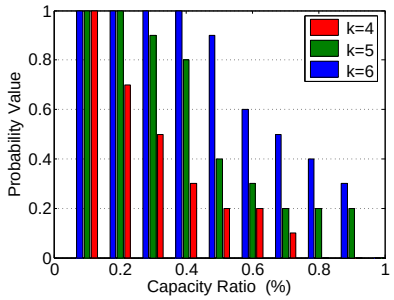

Fig. 5. The probability that the maximal MAL exists at the root.

structures. A given tree infrastructure is a computational-bottleneck if for any two-level, three-node subtree (shown in Figure 1 (b)), $N_l = min\{N_l, L_l\}$ and $N_r = min\{N_r, L_r\}$. The intuition behind the computational-bottleneck structure is that elasticity bottlenecks appear at the leaf nodes. A given tree is called a *fat-tree* [4] if for any two-level, three-node subtree, $L \geq L_l + L_r$. This fat-tree structure is frequently used in DCNs because upper links usually carry more traffic and a higher bandwidth must be used.

Theorem 5. *Given a binary tree that has a computational-bottleneck or is a fat-tree, the simple solution is optimal.*

Proof. When the tree is a computational-bottleneck structure, we can see that tree orientation does not change the maximum elasticity of the tree, so the simple solution works. In fact, link bandwidth does not play any role in the calculation. When the tree is a fat-tree, the bottleneck links are always at the lowest levels. Again, the tree orientation does not change the maximum elasticity. □

V. Simulation Comparisons

Basic setting: The DCN is modeled as a strict binary tree with levels $k = 4$, 5, and 6. The amount of nodes (i.e., physical machines) ranges from 5 to 30. The node space (i.e., physical machine capacity) is heterogeneous and ranges from 0 to 100 units. The unit of the resource is slotted, which can be easily interpreted to a real configuration. Each slot can hold one VM, and the bandwidth demand between per-pair of VMs is 1 Gbps. The link bandwidth is uniformly random. The ratio between the average node space and link bandwidth ranges from 0 to 1. For example, if the node space is 20 slots with 0.5 link ratio, the bandwidth capacity is 10 Gbps. In addition to the proposed scheduling algorithm, three baseline algorithms are used. (i). Equally Distributed Placement (EDP): the VMs are evenly assigned into the nodes in the tree. (ii). Proportion with Physical Machine Capacities (PPMC): the VMs are assigned into the nodes with proportional of the space. (iii). Proportion with Physical Link Capacities (PPLC): the VMs are assigned into the nodes with proportional of the bandwidth.

Analysis of the maximal MALs: Since the scales and capacities of the trees are different, the localities of the maximal MALs may also be different. As shown in Figure 5, we analyze the probability that the maximal MAL exists on the root. We find that the probability decreases as the size of the tree increases. This means that with the scaling of tree size, bandwidth becomes the main limitation of communication between PMs. The probability decreases with an increase in the ratio between the physical machines' capacities and the physical links' capacities. The center point gradually shifts downward from the root with the bandwidth limitation. Figure 6 presents the comparison of the performances of the simple and optimal solutions by calculating the mean value of different DCNs ($k = 4$, 5, and 6) under various capacity ratios ranging from 0 to 1. We have the following observations: (i). When the ratio between the physical machines' capacities and the physical links' capacities is low, meaning the physical links are not the bottleneck of the available resource, the values of the maximal MAL may be approximately equal in both the simple and optimal solutions, as shown in the three figures of Figure 6. (ii). When the size of the tree is scaling, the upper links may be the bottleneck for communication between physical machines. The locality of the maximal MAL may be shifted downward from the root, and the gap between the simple and optimal solutions increases with the scale of the tree. The admissible load of the optimal solution is 50% more than that of the simple solution, as shown in Figure 5 and Figure 6 (c).

Analysis of elasticity with placement under the maximal MAL: We place VMs based on Proportion with Physical Combinational Capacities (PPCC). Each virtual request iteratively places using the proportion of the bottleneck resource, which may be either physical machine capacities or physical link capacities. The value of the resource demand for each request ranges from $[0, 200]$. For each group, we do the placement based on the available resource and calculate the elasticities. We average the results of the placement 10 times for virtual requests with different algorithms. We have the following observations: (i). As shown in Figures 7 (a), (b), and (c), the elasticity grows with the scaling of the tree size. (ii). The elasticity depends on the capacity ratio. As the blue line shows in Figure 7, the combinational elasticities under the strategies are decreasing with the increasing ratio. We can have that when the ratio of the average node space to the average link bandwidth is lower, the bandwidth is not the bottleneck. (iii). The elasticity also depends on the placement strategy. As shown in the green lines in Figure 7, when the ratio becomes larger, the performance of the PPMC decreases. In contrast, the performance of the PPLC improves with an increase in the capacity ratio, which means the accuracy of the PPLC depends on the constraint of physical links. Our strategy has the best performance in elasticity compared to the baseline algorithms; it improves the resultant elasticity by 16.2%, 17.2%, and 11.9% under $k = 4$, 5, and 6, respectively.

VI. Related Work

Virtualization technology ensures application isolation and, at the same time, allows for utilization of the physical machine. Much work has been done in VM placement in cloud-based DCNs [1] with constraints that include power and performance [5], reliability [6], and traffic minimization [7]. [8] discussed some other practical factors in VM scheduling. Predictability is an important goal in designing efficient DCNs [9]. [10] uses empirical estimates of bandwidth in the placement design.

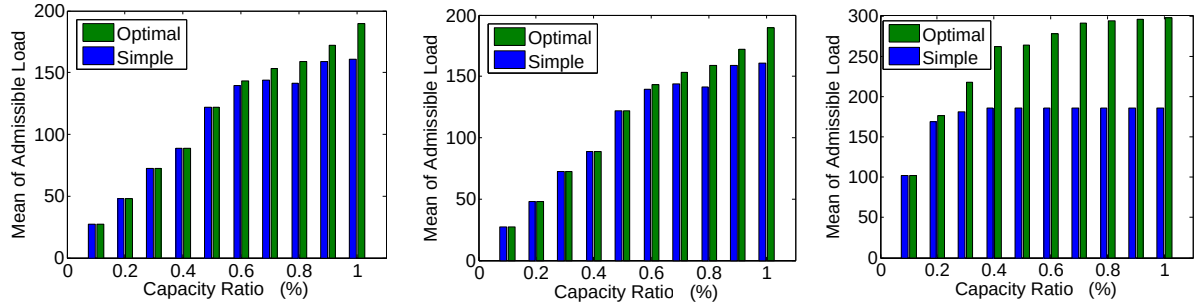

Fig. 6. Caparison of the elasticities of the simple and optimal solutions (a): $k = 4$, (b): $k = 5$, and (c): $k = 6$.

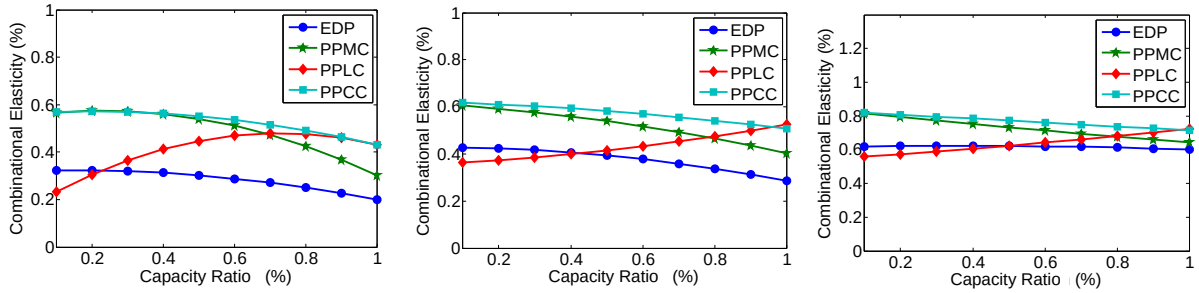

Fig. 7. Comparison of the elasticities between the simple and optimal solutions (a): $k = 4$, (b): $k = 5$, and (c): $k = 6$.

Elasticity has been considered one of the central attributes when estimation cannot be easily obtained [11, 12]. In cloud computing, elasticity is defined as the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner [13]. In [2], the authors consider elasticity-aware VM placement in a tree-based DCNs, taking both computation load and communication bandwidth into consideration. However, our proposed approach is the only one to achieve optimality when the tree is semi-homogeneous, i.e., the bandwidth of the links at the same level are the same, but ones at different levels are different. The scheme proposed in this paper extends the optimality to general tree structures.

VII. Conclusion

This paper proposes maximum elasticity scheduling that supports maximum future growth without resorting to task re-scheduling. It is based on an iterative abstraction that includes both maximum computation elasticity and maximum communication elasticity. Our insight into maximum communication elasticity stems from a special type of the hose model which determines the communication load based on the underlying computation load. Given a tree-structured DCN, we offer a distributed, optimal solution that computes the maximum admissible load and performs the maximum elastic scheduling of any admissible load. Experiments demonstrate the efficiency and effectiveness of our approach. The optimal solution can be easily extended to any k-nary tree structure. In our future work, we will explore the elasticity in multiple path routing, such as MPLS [14].

VIII. Acknowledgments

This research was supported in part by NSF and CSC grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, and CSC 20163100.

References

[1] Z. Á. Mann, "Allocation of virtual machines in cloud data centersa survey of problem models and optimization algorithms," *ACM Computing Surveys (CSUR)*, vol. 48, no. 1, 2015.

[2] K. Li, J. Wu, and A. Blaisse, "Elasticity-aware virtual machine placement for cloud datacenters," in *IEEE 2nd International Conference on Cloud Networking (CloudNet)*, 2013.

[3] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, "A flexible model for resource management in virtual private networks," in *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, 1999.

[4] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. 100, no. 10, 1985.

[5] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Springer Cluster Computing*, vol. 12, no. 1, 2009.

[6] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable virtual machine placement and routing in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, 2017.

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

[8] F. Xu, F. Liu, H. Jin, and A. V. Vasilakos, "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions," *Proceedings of the IEEE*, vol. 102, no. 1, 2014.

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

[10] R. Wang, J. A. Wickboldt, R. P. Esteves, L. Shi, B. Jennings, and L. Z. Granville, "Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in datacenters," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, 2016.

[11] D. Plummer, Lummer, D. Smith, T. Bittman, D. Cear-Ley, D. Cappuccio, D. Scott, R. Kumar, and B. Robertson, "Study: Five refining attributes of public and private cloud computing," http://www.gartner.com/DisplayDocument, Tech. Rep., 2009.

[12] S. Lu, Z. Fang, and J. Wu, "Elastic scaling of virtual clusters in cloud data center networks," in *IEEE IPCCC*, 2017.

[13] N. R. Herbst, S. Kounev, and R. H. Reussner, "Elasticity in cloud computing: What it is, and what it is not." in *ICAC*, vol. 13, no. 10. USENIX, 2013.

[14] B. S. Davie and Y. Rekhter, *MPLS: technology and applications*. Morgan Kaufmann Publishers, 2000.