Lu SB, Wu J, Zheng HY *et al.* On maximum elastic scheduling in cloud-based data center networks for virtual machines with the hose model. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 34(1): 185–206 Jan. 2019. DOI 10.1007/s11390-019-1890-3

On Maximum Elastic Scheduling in Cloud-Based Data Center Networks for Virtual Machines with the Hose Model

Shuai-Bing Lu[1,2], *Student Member*, *IEEE*, Jie Wu[2,*], *Fellow*, *IEEE*, Huan-Yang Zheng[2], and Zhi-Yi Fang[1]

[1] *College of Computer Science and Technology, Jilin University, Changchun 13012, China*
[2] *Department of Computer and Information Sciences, Temple University, Philadelphia, PA19122, U.S.A.*

E-mail: lushuaibing11@163.com; jiewu@temple.edu; huanyang.zheng@gmail.com; fangzy@jlu.edu.cn

Received July 15, 2018; revised August 30, 2018.

Abstract With the growing popularity of cloud-based data center networks (DCNs), task resource allocation has become more and more important to the efficient use of resource in 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 runtime when the load grows, we introduce maximum elasticity scheduling, which has the maximum growth potential subject to the node and link capacities. This paper aims to find the schedule with the maximum elasticity across nodes and links. We first propose a distributed linear solution based on message passing, and we discuss several properties and extensions of the model. Based on the assumptions and conclusions, we extend it to the multiple paths case with a fat tree DCN, and discuss the optimal solution for computing the MAL with both computation and communication constraints. After that, we present the provision scheme with the maximum elasticity for the VMs, which comes with provable optimality guarantee for a fixed flow scheduling strategy in a fat tree DCN. We conduct the evaluations on our testbed and present various simulation results by comparing the proposed maximum elastic scheduling schemes with other methods. Extensive simulations validate the effectiveness of the proposed policies, and the results are shown from different perspectives to provide solutions based on our research.

Keywords data center network (DCN), cloud, distributed algorithm, elasticity, hose model, optimization

1 Introduction

With the increasing popularity of cloud-based data center networks (DCNs), task resource allocation has become more and more important to the efficient use of resource in 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[1,2]. This paper is focused on a new quality of service (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 [3], but its optimal solution is limited to a semi-homogeneous tree structure. In this paper, we extend the optimal solutions to two general cases which can be used on the heterogeneous tree (single path) and the fat tree (multiple paths).

1.1 Motivational Example

We first model the network as a tree-structured T in a typical DCN, as shown in Fig.1. Each leaf node is a physical machine and each internal node is a switch.

Regular Paper

A preliminary version of the paper was published in the Proceedings of ICC 2018.

This research was supported in part by the National Science Foundation (NSF) of United States under Grant Nos. CNS 1757533, CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, IIP 1439672, and CSC 20163100.

*Corresponding Author

A load at a leaf node is called a computation load and determines the communication load. We use the hose model[4] for communication where each node has an aggregated performance guarantee corresponding to the set of all the other nodes.

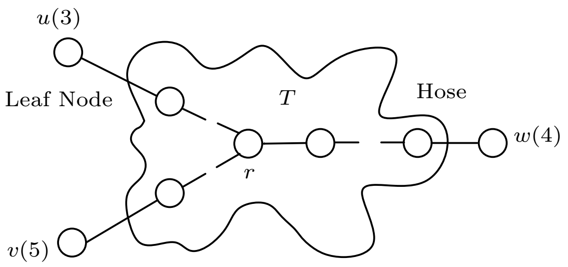


Fig.1. Example of a cut in a tree under the hose model.

Fig.2(b) shows a two-level, three-node binary tree where each PM (a leaf node) is represented by a slotted rectangle (e.g., VM slot or computation load), 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 and the right links are the same: it is the lesser of the assigned VMs in the two leaf nodes multiplied by B based on the hose model. This is analogous to the maximum flow in a cut as shown in Fig.2(a). If the left leaf node of Fig.2(b) is assigned two VMs of one part of the cut and the right leaf node is assigned three VMs of the other part, the communication load is $\min\{2B, 3B\} = 2B$ based on the hose model. If the left leaf node is assigned one VM and the right four VMs, the communication load is $\min\{1B, 4B\} = B$. When the left and the right node loads are doubled in Fig.2(b), the communication load becomes $4B$. This load can still be handled by the left and the right links. However, if we allocate the computation load based on the node capacity, the corresponding communication load is $\min\{5B, 6B\} = 5B$, which exceeds $4B$ in the available bandwidth of the left link.

Using the hose model, we study two provisioning problems.

1) Given a graph T with available capacities of nodes and links, 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?

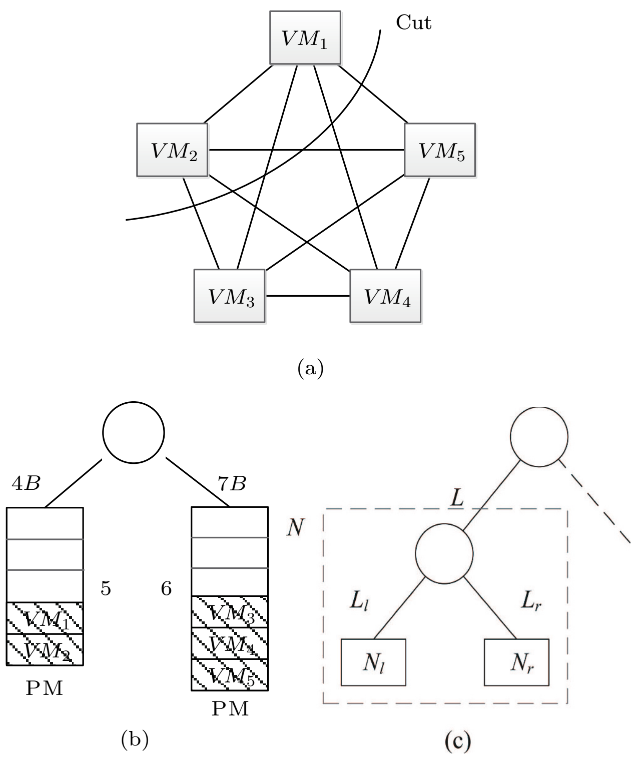


Fig.2. VM scheduling as a (a) cut, (b) virtual node, and (c) aggregation tree.

The maximum admissible load (MAL) is defined as the maximum number of virtual machines (VMs) whose computation and communication load can be held underlying the tree-structured DCN T. The first problem is analogous to a special utility allocation problem: how to lay out cables for all households in a community for a telecom company. The cable connection is organized in a tree structure to minimize the layout cost. Each section of the cable line has a capacity. Each household has an occupancy limit. What are the maximum total occupancy limit and the actual assignment at each house so that cable lines can support the bandwidth of all possible simultaneous pairwise telephone conversations? In Fig.1, suppose the numbers of occupants living at households u, v, and w are 3, 5, and 4, respectively. One possible simultaneous pairing is the following: three pairs between u and v, two pairs between v and w, and one pair within w. In this case, the bandwidth requirement for the cut link is 2 (assuming that each pair of telephone conversations takes 1 unit of bandwidth). Another pairing can be two pairs between u and v, one pair between u and w, and three pairs between v and w. In this case, the bandwidth requirement for the cut link is 4. Since there is no constraint on the link capacity, the maximum total occupancies of users in these two cases are both equal to $12 = 3 + 5 + 4$. It means that we can guarantee six simultaneous pairwise telephone conversations for communication, where each location (u, v, or w) has an aggregated performance guarantee corresponding to the set of all the other locations. However, if we suppose that the link capacities of u, v, and w are 3, 5, and 2 respectively, then although there are 12 available slots, the maximum total occupancy of users is only $10 = 3 + 5 + 2$ due to the link capacity constraint of w. In this case, the first configuration can support 12 slots, but the second configuration can support 10. Note that, under the same path capacities, when the numbers of occupants living at households u, v, and w are changed to 1, 1, and 4 respectively, the maximum total occupancy of users is $6 = 1 + 1 + 4$ when the path capacity from w to r is 2. That is the maximum number of pairwise telephone conversations that go through the path from w to r when its capacity is 2 under any configuration.

The optimal schedule for the second problem is called the schedule with the maximum elasticity. As shown in Fig.2(b), the MAL is 10, with four VMs (also loads) assigned to the left leaf node and six loads to the right leaf node. Both the left link and the right leaf node reach the 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 two loads to the left leaf node and three loads to the right leaf node, as shown in Fig.2(b). The maximum elasticity is 100%. That is, each side can be doubled without violating the node or link capacities. 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 simple and efficient solutions, similar to the Bellman-Ford solution of the shortest path problem.

The complexities of our solutions are linear to n where n is the number of leaf nodes in the tree in terms of both computation and communication complexities.

1.2 Our Contributions

• We first introduce the concept of 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 as a priori). This aggregation tree gives us a simple iterative solution that abstracts each two-level, three-node branch (such as the one in Fig.2(b)) into one virtual node. The abstraction is a bottom-up aggregation process that determines the MAL at the root of the tree, and 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 complexities are linear to n where n is the number of leaf nodes in the tree.

• Based on the assumptions and conclusions above, we extend our model from the tree structure with a single path to the fat tree DCN with multiple paths. We also extend our optimal solution on computing the MAL of a fat tree and present a corresponding placement scheme with maximum elasticity.

• We conduct the evaluations on our testbed, and present various simulation results by comparing the proposed maximum elastic scheduling schemes with other methods. Extensive simulations validate the effectiveness of the proposed policies. The results are shown from different perspectives to provide conclusions.

The remainder of our paper is organized as follows. Section 2 introduces the elastic scheduling of VMs for the tree-structured DCN. Section 3 presents the elastic scheduling of VMs for the fat tree DCN. Section 4 shows our experiments and simulation results. Section 5 overviews the related work. Finally, the paper is concluded in Section 6.

2 Maximum Elastic Scheduling of VMs for Tree DCN

In this section, we present the maximum elastic scheduling of VMs for the tree DCN. We first show that the maximum elasticity problem can be solved by the classic LP, and we discuss the inefficiency of this solution. Then, we study a simple distributed message passing linear solution with a tree-based topology and discuss several properties of the model. Based on that, we further propose a distributed optimal solution that uses only three copies of the simple solution to compute different orientations of a tree with different roots.

2.1 Basic Solution Based on LP

We first use an LP approach to maximize the elasticity in trees. Let N_i and x_i denote the maximum space and the used space of the i-th node, respectively. Let L_j and y_j denote the maximum bandwidth and the used bandwidth of the j-th link, respectively. A boolean variable, μ_{ij} is used to indicate whether the i-th node belongs to the subtree of the j-th link. We have the following problem formulation:

$$\text{maximize} \quad e \tag{1}$$

$$\text{s.t.} \quad e \leqslant \min_i (1 - \frac{x_i}{N_i}) \text{ and } x_i \leqslant N_i \text{ for } \forall i, \tag{2}$$

$$e \leqslant \min_j (1 - \frac{y_j}{L_j}) \text{ and } y_j \leqslant L_j \text{ for } \forall j, \tag{3}$$

$$y_j = \min(\sum_i \mu_{ij} x_i, \sum_i (1 - \mu_{ij}) x_i) \text{ for } \forall j. \tag{4}$$

Here, e denotes the elasticity for both nodes and links. (1) shows the objective of maximizing the elasticity. (2) and (3) are constraints on nodes and links, respectively. (4) computes the bandwidth consumption based on the hose model. This formulation is not linear; however, it can be converted to an LP formulation as follows:

$$\text{maximize} \quad e \tag{5}$$

$$\text{s.t.} \quad e \leqslant 1 - \frac{x_i}{N_i} \text{ and } x_i \leqslant N_i \text{ for } \forall i, \tag{6}$$

$$e \leqslant 1 - \frac{y_j}{L_j} \text{ and } y_j \leqslant L_j \text{ for } \forall j, \tag{7}$$

$$y_j \leqslant \sum_i \mu_{ij} x_i \text{ and } y_j \leqslant \sum_i (1 - \mu_{ij}) x_i \text{ for } \forall j. \tag{8}$$

(5) is the same as (1). (6) converts (2) by separating the minimum constraint of each node. Similarly, (7) converts (3) by separating the minimum constraint of each link. (8) converts (4) by separating the minimum constraint to two parts. Let n denote the number of leaf nodes in the tree, and then the number of links in a full binary tree is $2n - 2$. (6) contains $2n$ constraints, (7) includes $4n - 4$ constraints, and (8) includes $4n - 4$ constraints. In total, the LP formulation has $1 + n + 2n - 2 = 3n - 1$ variables and $10n - 8$ constraints. In general, LP solutions require a complexity of a polynomial function of n. Using the latest result such as [5], it is still unclear that the LP formulation of our problem can be reduced to $\Theta(n)$. It means that our problem cannot be efficiently solved by the simplex method or the eclipse method. Such inefficiency motivates us to find other direct solutions for elasticity maximization in Subsection 2.2 and Subsection 2.3. The proposed direct optimal solution will provide more insights and it has the linear complexity in terms of n.

2.2 A Simple Up-Down Solution

We first propose a simple distributed message passing linear solution with a tree-based topology and discuss several properties of the model. Given a binary tree (or simply a tree) T with available node space and link bandwidth, our objective is to find an admissible scheduling with maximum elasticity under the hose model. A schedule is admissible if the corresponding scheduling of tasks to leaf nodes does not exceed the available node capacity. In addition, no communication link exceeds its available link bandwidth under the hose model. By the maximum elasticity, we mean that a schedule with the maximum uniform growth at each leaf node does not violate the node and link capacities. In fact, the maximum elasticity includes both the maximum computation elasticity at leaf nodes and the maximum communication elasticity at links.

2.2.1 *Bottom-Up Abstraction*

The simple solution iteratively abstracts the given tree in a bottom-up manner. As shown in Fig.2(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 children of the internal node. At the bottom level of the tree, a virtual node is a leaf node. At all the 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 the right virtual nodes, respectively, as shown in Fig.2(c). The middle node in the abstraction serves two purposes. First, the middle node is a "connector" between two branches. A cut to either the left or the right link has a value of $\min\{\min\{N_l, L_l\}, \min\{N_r, L_r\}\}$. This is the communication load between the left and the right branches. This load is clearly not more than L_l and L_r. In Fig.2, the link cut value is $\min\{\min\{5, 4\}, \min\{6, 7\}\} = 4$, which is not more than $L_l = 4$ and $L_r = 7$. Second, the middle node is also an "aggregator" for two branches. It is used to calculate the value of the cut at an upper level of the tree. The capacity of the aggregation virtual node is determined as follows: $N = \min\{N_l, L_l\} + \min\{N_r, L_r\}$. 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. Fig.3 shows this process on a three-level full binary tree. The load of the left virtual node in Fig.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 Fig.3(c) is decided by abstraction from Fig.3(b). In this case, the MAL is $16 = \min\{10, 11\} + \min\{6, 6\}$.

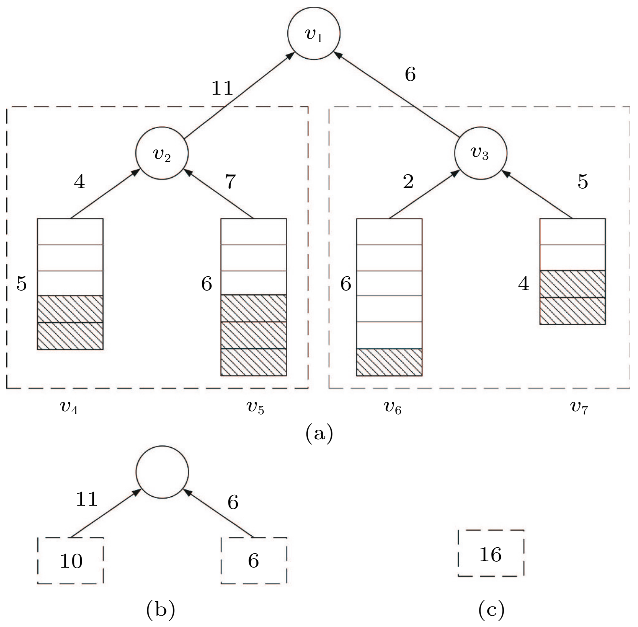


Fig.3. MAL calculation of a (a) binary tree, (b) three-node branch, and (c) virtual node.

2.2.2 *Top-Down Schedule*

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 Fig.3. Supposing that N is a given load that is not more than the MAL, we partition the load into two parts. The left and the 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 Fig.3, let 8 be a given load that is less than the MAL 16. Based on Fig.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 continues at the next level. The final assignment is in Fig.3(a), and the load assignment is shown as shaded slots. This schedule corresponds to the one with the maximum elasticity, which is 100% of the current load. The load assignments to v_6 and v_7 are 1 and 2, respectively. These are not proportional to their available node spaces. The hose model provides aggregate performance requirements for each hose connected to an endpoint. On the other hand, the pipe model offers a specific performance requirement for each pair of endpoints. The provisioning of a hose structure can be interpreted as many different pipe structures. Using Fig.3 as an example, we suppose that each pipe consumes one unit of space and bandwidth. The hose structure of Fig.3 can be viewed as two pipes for (v_4, v_5), one pipe for (v_5, v_7), and one pipe for (v_6, v_7). Another possible pipe structure can be one pipe for (v_4, v_5), two pipes for (v_5, v_7), and one pipe for (v_5, v_6). Different pipe configurations generate different communication loads at links.

2.3 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. Fig.4 shows a slightly revised version of the example in Fig.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 Fig.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.

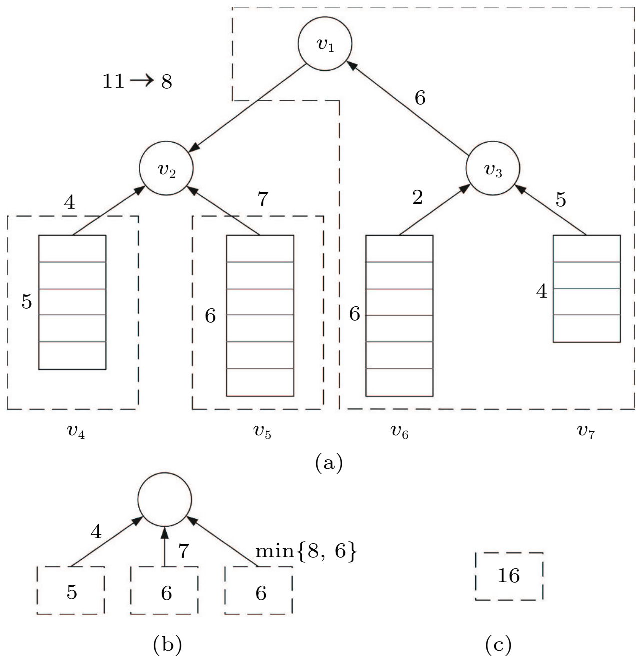


Fig.4. Optimal solution through different orientations of aggregation at one node in a (a) binary tree, (b) three-node branch, and (c) virtual node.

2.3.1 *Aggregation Tree*

Now we introduce a new notation called aggregation tree based on the hose model. 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 the 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 less one of the two computation loads that come from both sides of the cut. In Fig.5, the root is represented as a double cycle supposing 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.

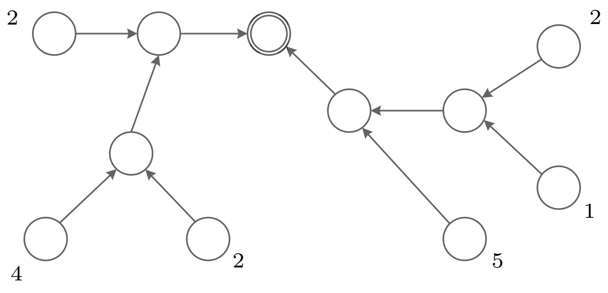


Fig.5. Aggregation tree based on the hose model.

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 does not have more than 50% of the total workload, it points towards its neighbor. In addition, its workload will be added to its 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 a case where the leaf node has exactly 50% of the workload, then the root occurs when the ID is larger. It is clear that two roots are impossible due to the majority workload requirement. □

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 aggregation tree. In Fig.5, the root is represented as a double cycle supposing that the link bandwidth is infinity. The workload from the left branch pointing towards the root is $2 + 4 + 2 = 8$, and the workload from the right branch is $2+1+5 = 8$.

2.3.2 *Elastic Schedule Based on Aggregation Tree*

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 toward the selected root for aggregation, as shown in Fig.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 Fig.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. Three cases are shown in Fig.6. 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 branches and its local computation load.

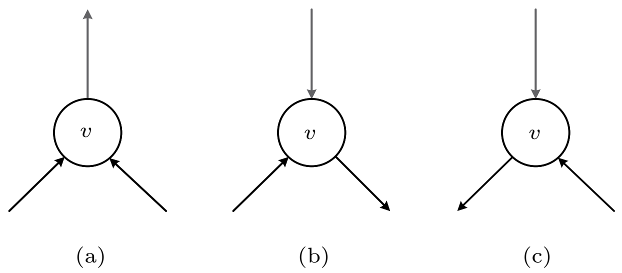


Fig.6. Optimal solution through a different orientation of aggregation at (a) root, (b) right branch, and (c) left branch.

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 not more than L_1). In Fig.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 not more than its available communication bandwidth. A root selection example for a two-level, three-node tree is illustrated in Fig.2(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 as a branch with infinite bandwidth. When N_l, N_r, L_l, and L_r are 8, 4, 5, and 4 respectively, the MALs 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%.

In the following optimal solution, each leaf node initiates its calculation by sending the available load to the connected internal node. Table 1 shows an example of a step-by-step calculation of the MAL for each node in Fig.4. The final MAL is the maximum one among the MALs calculated at all internal nodes. Once the MAL is determined with the selected root node, the schedule with the maximum elasticity 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 Fig.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 Fig.4(b). The left, middle, and right branches are assigned $8\times4/16 = 2$, $8\times6/16 = 3$, and $8\times6/16 = 3$, 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 $(16 - 8)/8 \times 100 = 100\%$.

2.4 Properties and Extensions

This subsection studies several properties and trivial extensions of the optimal solution and the simple solution for the tree DCN (Algorithm 1).

2.4.1 *Properties*

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. Considering each two-level, three-node tree at the bottom two levels of the tree, we first show that $\min\{N_l, L_l\} + \min\{N_r, L_r\}$

Table 1. Step-by-Step Calculation of MALs for the Example of Fig.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	$\min\{10,8\}+$ $\min\{10,8\}+$ $\min\{6,6\} = 14$	$\min\{5,4\}+$ $\min\{6,7\}+$ $\min\{8,6\} = \mathbf{16}$	$\min\{6,2\}+$ $\min\{4,5\}+$ $\min\{8,6\} = 12$	$\min\{12,4\}+$ $\min\{12,4\}+$ $\min\{5,\infty\} = 9$	$\min\{10,7\}+$ $\min\{10,7\}+$ $\min\{6,\infty\} = 13$	$\min\{10,2\}+$ $\min\{10,2\}+$ $\min\{6,\infty\} = 8$	$\min\{8,5\}+$ $\min\{8,5\}+$ $\min\{4,\infty\} = 9$

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 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. □

Algorithm 1 . Optimal Solution for Tree DCN

```
Input: tree DCN;
Output: MAL for the tree;
    (at a leaf node with node capacity $N$)
 1: Send its load to the connected internal node.
    /*Upon receiving a virtual load $N_1$ from the only branch with available link bandwidth $L_1$*/
 2: 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 $L_i$, $i : 1, 2$*/
 3: Send virtual load $\min\{N_i, L_i\}$ to the other branch.
    /*Upon receiving virtual load, $N_1$ and $N_2$, from two branches*/
 4: 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 $i, j (\neq i) : 1, 2, 3$*/
 5: 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*/
 6: Calculate its MAL: $\min\{N_1, L_1\} + \min\{N_2, L_2\} + \min\{N_3, L_3\}$.
```

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 Theorem 3 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 the right branches is optimal. The fact that the proportions are determined by the maximum load of both the left and the 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 a total of n times computation. In addition, each node calculates its MAL once, which means $2n-1$ times computation in the binary tree in total. Therefore, the computation complexity is $5(n-1)$. Each leaf node communicates once for a total of n times computation. Each link that connects two internal nodes communicates twice (once in each direction) to pass along virtual load information for a total of $2(n-2)$ times communication. Each leaf node receives virtual load information once from an internal node for a total of n times communication. □

When we consider elasticity, the bottleneck must be either in the link or in 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 structures. A given tree infrastructure is a computational-bottleneck if for any two-level, three-node subtree (shown in Fig.2(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[6] if for any two-level, three-node subtree, $L \geqslant L_l + L_r$. This fat tree structure is frequently used in DCNs because upper links usually carry more traffic and a higher bandwidth must be used. In order to distinguish with the fat tree[7] we use in Section 3, we use fat tree* to denote the two-level topology in [6].

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; therefore 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. □

2.4.2 Extensions

General Trees. The optimal solution can be easily extended to any k-ary tree structure with up to $k+1$ branches at an internal node. The key difference is that each internal node needs to keep track of the virtual load value for each branch. Whenever a node receives a virtual load from all branches except itself, it determines the aggregate value of these virtual loads and sends the value to the branch that is not involved in the aggregation process. The aggregation process is the same as the proposed method for an abstraction of a two-level, $(k+1)$-node subtree. In Section 3, we focus on extension to other structures, fat tree[7] (see the example in Fig.7, which includes an aggregation layer consisting of the pods and a core layer). An interesting future study would be the extension to an infrastructure with a regular graph. We can also explore elasticity in multiple path routings like multi-protocol label switching (MPLS)[8].

Developing Other Communication Load Models. To generalize the communication load models, we set $L = f(N)$, where $f(\cdot)$ is a constant multiplier with a constant coefficient c. For example, $L = 2N$ when $c = 2$. To avoid remapping f, we first scale down the available link bandwidth by a factor of c. Thus, the same optimal solution can be applied. Our approach cannot be directly applied when the mapping function is non-linear because the total communication load generated depends on the way the computation load is partitioned, which is the case when f is a quadratic function. For the co-existence of multiple requests, we consider applying two cases to our approach. If each request is independent, then one distinct virtual private network (VPN) is used for each request. For multiple requests that belong to the same application, no new VPN is needed. Instead, both the existing and the new workloads are combined for distribution as a virtual load, and then the optimal virtual load for each leaf node is determined based on the proposed method. The actual new load assignment to each leaf node is calculated by abstracting the existing load for that leaf node.

Special Configurations for Efficiency Gain. When we consider elasticity, the bottlenecks must be either in links or in nodes in terms of capabilities of growth. In this thrust, we consider special situations under which the simple solution is optimal. The focus of this thrust will be the structure of the network and the capacity of the load/bandwidth. Next, we consider two special situations under which the simple solution is optimal. Let us first introduce two special structures (refer to Fig.2(a) for an illustration). A given tree infrastructure is a computational-bottleneck if for any two-level, three-node subtree (shown in Fig.2(c)), $N_l = \min\{N_l, L_l\}$ and $N_r = \min\{N_r, L_r\}$. The intuition behinds the computational bottleneck structure is that elasticity bottlenecks appear at the leaf nodes. For example, for any two-level, three-node subtree in a fat tree*[6], $L \geqslant L_l + L_r$, as shown in Theorem 5.

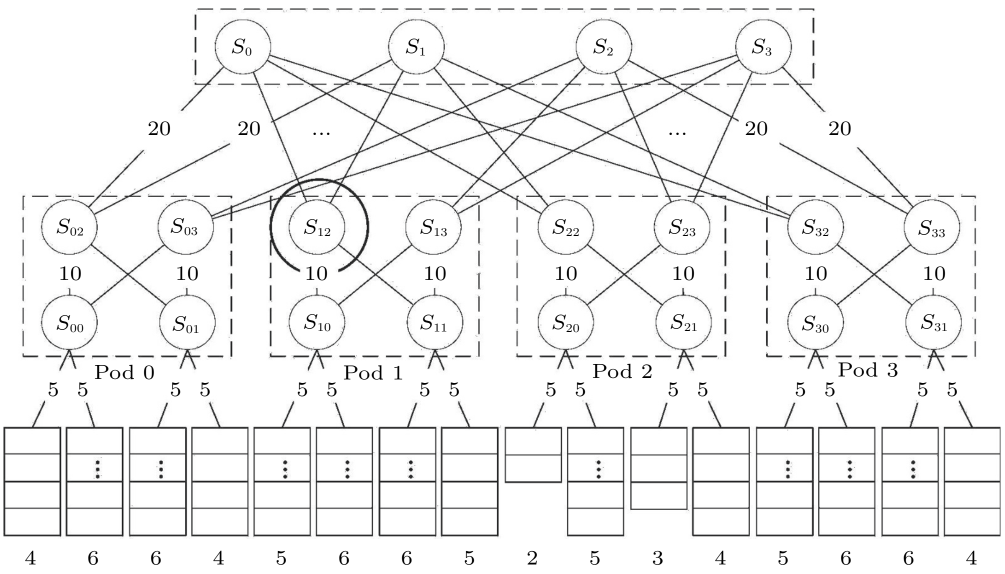


Fig.7. Example of a fat tree.

3 Maximum Elastic Scheduling of VMs for Fat Tree DCN

In this section, we extend our optimal solution to the fat tree DCN[7,9][1]. The aggregation process is similar to the proposed method for an abstraction of a tree-structured DCN. As we discussed in Subsection 2.4, the optimal solution can be easily extended to any tree structure. The key difference is that each internal node needs to keep track of the virtual load value for each branch. Whenever a node receives a virtual load from all other branches except itself, it determines the aggregate value of these virtual loads and sends the value to the branch that is not involved in the aggregation process. The difference during the VMs provisioning process in fat tree is in determining how to evaluate the MAL value for both the occupancy and the actual assignment under the multiple path routing. Linear programming is the most typical approach to calculating the value of MAL. Since there are many variables in the fat tree DCN, either the PM or the link capacity constraints may be the bottleneck. When the construction of the DCN is extremely large-scale, using linear programming to obtain the optimal solution is slow and inefficient.

3.1 Optimal Solution of the Fat Tree DCN

We consider the VMs provisioning problem based on the hose model in the fat tree DCN and divide this problem into two steps. 1) We first consider provisioning the MAL for the fat tree DCN based on the aggregation method of different orientations. 2) Next, we find the provisioning for the VMs with the maximum elasticity, which comes with provable optimality guarantees for a fixed flow scheduling strategy in the fat tree DCN. Our objective is to find an admissible provisioning for the VMs with maximum elasticity in the fat tree DCN at the allocated PMs and links. Suppose that the flow scheduling rule for each set of VMs is fixed; therefore the capacities of links are only related to the distribution of VMs on the PMs.

3.1.1 Aggregation of the Fat Tree DCN

In this subsection, we extend our optimal solution on calculating the MAL of the fat tree DCN. Given a fat tree, our objective is to find the MAL without the computation and communication capacity constraints. The main idea of this solution is to do the iterative aggregation on different orientations and select the maximum one as the MAL. We first do the aggregation bottom-up to send information to the upper switches and use the received virtual loads to determine the orientation. Since the orientation of each link in the hose model is determined by the less computation load that comes from both sides of the cut, we use the orientation to select the root of the fat tree. We use r to denote the real root in the fat tree DCN. If the abstraction value of V_0 is larger than V_1, then the orientation would be from right to left and root r would exist on the left side (S_{i2}). Otherwise, it may be on the right side (S_{i3}). We take the whole fat tree as the input, which contains the computation and communication capacities for both PMs and links. The output is the MAL for the fat tree, and the fat tree is constructed by two types of nodes, PMs and physical switches (the edge switches, aggregation switches, and core switches). Each switch with m branches has m orientations, and the orientation of each branch points towards the selected root r for aggregation. The abstraction of the branches under one switch is defined as one virtual load V_{ij}. m-branch node receives virtual load information from $m-1$ branches, and it passes the information on to the connected node in the remaining branch.

In lines 1–3 of Algorithm 2, we start to do the aggregation from PMs. The MAL at a PM is the summation of its branches and its local computation load. There are $m/2-1$ branches for each PM in the fat tree, and each PM receives a virtual load V_{ij} from the only branch with available upper link bandwidth L_{ij}. The MAL at one PM is $\min\{N_{ij},\infty\}+\min\{V_{ij},L_{ij}\}$. In lines 4–7, we calculate the MAL for the edge switch. The MAL at an edge switch is the summation of its branches and its virtual loads. The edge switch upon receiving virtual loads from $m/2$ branches with available lower links' bandwidth, and each of the edge switches sends the virtual load $\min\{V_{ij},L_{ij}\}$ to other branches. Since each switch has m ports, each of them receives virtual loads from the other $m/2$ branches with available upper links' bandwidth. The MAL at one edge switch would be (9).

$$\sum_{j=1}^{j=m/2}\min\{N_{ij},L_{ij}\}+\sum_{j=m}^{j=2m-1}\min\{V_{ij},L_{ij}\}. \quad (9)$$

In lines 8–11, we calculate the MAL for the aggregation layer, which is the same process as for the edge

[1]This fat tree DCN is different from the fat tree* discussed in Section 2. For historical reason, this phrase fat tree is used to represent two different structures.

layer. The difference lies in the adjacent points of the aggregation layer, which are all switches; thus all the information they receive is in the form of virtual loads. The MAL at one edge switch is $\sum_{i=1}^{i=m}\min\{V_{ij}, L_{ij}\}$. In lines 12–14, we calculate the MAL for core switch. Since each core switch has m branches, upon receiving virtual loads from these branches with available lower links' bandwidth, it will send virtual load $\min\{V_{ij}, L_{ij}\}$ to other branches. The MAL at one core switch is also $\sum_{i=1}^{i=m}\min\{V_{ij}, L_{ij}\}$. The MAL of the fat tree may be located on either the PM or the switch node with the maximum value. Since communication in this paper uses the hose model, the location of the final MAL depends on the link orientation.

Algorithm 2 . Optimal Solution for Fat Tree DCN

```
Input: fat tree DCN;
Output: MAL for the fat tree;
1: Send its load to the connected edge switch;
2: Calculate its MAL:
   $\min\{N_{ij}, \infty\} + \min\{V_{ij}, L_{ij}\}$;
   /*Edge switch with $m$ branches*/
   /*Upon receiving virtual loads $V_{ij}$ from $m/2$ branches with
   available lower links' bandwidth $L_{ij}$, $j \in [0, m/2]$;*/
3: Send virtual load $\min\{V_{ij}, L_{ij}\}$ to other branches; /*Upon
   receiving virtual loads $V_{ij}$ from $m/2$ branches with available
   upper links' bandwidth $L_{ij}$, $j \in [m, 2m-1]$;*/
4: Calculate its MAL:
   $\sum_{j=1}^{j=\frac{m}{2}}\min\{N_{ij}, L_{ij}\}+\sum_{j=m}^{j=2m-1}\min\{V_{ij}, L_{ij}\}$;
   /*Aggregation switch with $m$ branches*/
   /*Upon receiving virtual loads $V_{ij}$ from $m/2$ branches with
   available lower links' bandwidth $L_{ij}$, $j \in [0, m/2]$;*/
5: Send virtual load $\min\{V_{ij}, L_{ij}\}$ to other branches; /*Upon
   receiving virtual loads $V_{ij}$ from $m/2$ branches with available
   upper links' bandwidth $L_{ij}$, $j \in [m, 2m-1]$;*/
6: Calculate its MAL: $\sum_{i=1}^{i=m}\min\{V_{ij}, L_{ij}\}$;
   /* Core switch with $m$ branches*/
   /*Upon receiving virtual loads $V_{ij}$ from $m$ branches with
   available lower links' bandwidth $L_{ij}$, $j \in [0, m]$;*/
7: Send virtual load $\min\{V_{ij}, L_{ij}\}$ to other branches;
8: Calculate its MAL: $\sum_{i=1}^{i=m}\min\{V_{ij}, L_{ij}\}$.
```

We consider an example to illustrate our solution in Fig.7, and the conversion is shown in Fig.8. We do the calculation for the MALs in Fig.7 step by step following the process of the optimal solution for the fat tree DCN.

• *Step* 1. Send the computation capacities to PMs, e.g., send 4 to N_{00}, and send 6 to N_{01}.

• *Step* 2. Upon receiving the virtual loads from leaf nodes, send them to the edge switches, e.g., send $\min\{4, 5\} + \min\{6, 5\} = 9$ to S_{00}, and send $\min\{6, 5\} + \min\{4, 5\} = 9$ to S_{01}.

• *Step* 3. Upon receiving the virtual loads from the edge switches, send them to the aggregation switches, e.g., send $\min\{9, 10\}+\min\{9, 10\} = 18$ to S_{02}, and send $\min\{9, 10\} + \min\{9, 10\} = 18$ to S_{03}.

• *Step* 4. Upon receiving the virtual loads, send them to the core switches, e.g., send $\min\{18, 20\} + \min\{19, 20\} + \min\{14, 20\} + \min\{19, 20\} = 70$ to S_0.

• *Step* 5. Send virtual loads to other branches (aggregation switch) from up to bottom, e.g., send S_0 and S_1 to S_{02}. $S_0 = S_1 = \min\{19, 20\} + \min\{14, 20\} + \min\{19, 20\} = 52$. Combine the virtual loads based on the equal split flow schedule, where $\min\{S_0, 20\} + \min\{S_1, 20\} = 40$.

• *Step* 6. Send virtual loads to other branches (edge switch) from up to bottom, e.g., send $\min\{52, 20\} + \min\{52, 20\} + \min\{10, 9\} = 49$ to S_{00}.

• *Step* 7. Send virtual loads to other branches (PMs), e.g., send $\min\{49, 10\} + \min\{49, 10\} + \min\{6, 5\} = 25$ to N_{00}.

• *Step* 8. Calculate the MAL at each node and the maximum MAL among the MALs calculated at all internal nodes is S_{12}. Thus, the MAL is $\min\{51, 20\} + \min\{51, 20\} + \min\{9, 10\} + \min\{10, 10\} = 59$.

3.1.2 Elastic Schedule for VMs in Fat Tree

In this subsection, we propose a centralized provisioning scheme for VMs with maximum elasticity based on the MAL, which is called Proportion with Physical Combinational Capacities (PPCC). This comes with provable optimality guarantees for a fixed flow scheduling strategy in the fat tree.

We take a tuple as our input, which includes both the computation demand N and the communication demand B. In addition, we take the occupation state of the fat tree DCN as our output. Let R denote the value of the MAL. In line 1 of Algorithm 3, we initialize the value of R by using the optimal solution for the fat tree, as shown in Algorithm 2. Since the communication demand B of each VM is unique, we compare the number of VMs N of the set C with the MAL in line 2. If the total number of available physical resources of fat tree cannot satisfy the computation demand ($N > R$), we will reject it directly. Otherwise, we start to place N VMs. In line 3, we first find and determine the location of the root r for the fat tree DCN. Based on the functions of nodes in the DCN, we divide them into two types of root: root of a server node and root of a switch node. If r is a server node, we first allocate $N \times N_{ij}/R$ VMs into root r. After that, we allocate the remaining $N - N \times N_{ij}/R$ VMs based on the proportion of virtual load at each branch of the upper edge switch of root r, iteratively. If r is a switch node, then we determine the allocation of the VMs based on the proportion of virtual load at each branch from the root r to the bottom,

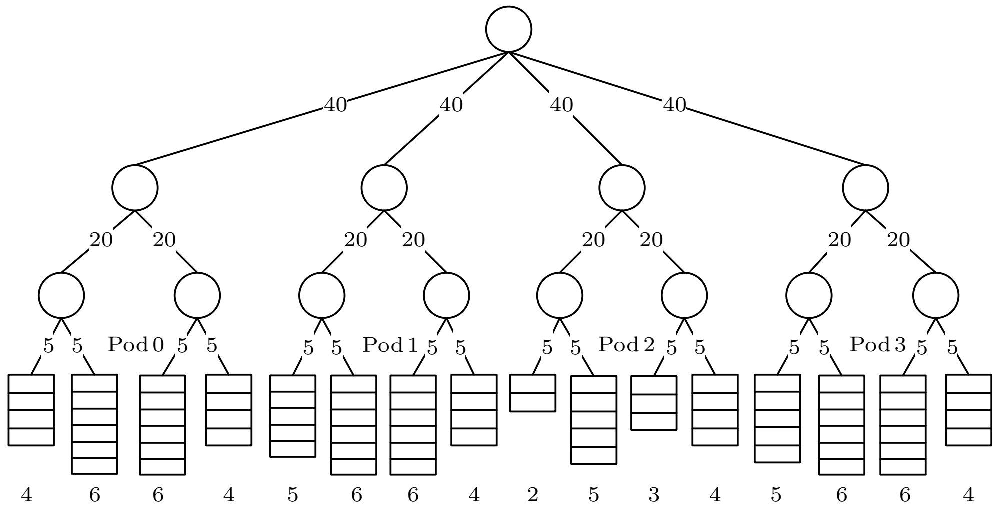


Fig.8. Conversion of example in Fig.7.

iteratively. Once the allocation of VMs is determined, the communication demands of placed VMs are split into paths based on the previously fixed flow schedule rule in line 9. After that, we update the value of R according to the distribution of existing loads in line 10.

Algorithm 3 . Proportion with Physical Combinational Capacities (PPCC)

```
Input:  a tuple of demand $(N, B)$;
Output: DCN occupation state for VMs;
 1: Calculate the MAL $\rightarrow R$;
 2: if $N \leqslant R$ then
 3:     Determine the location of root $r$ in the fat tree DCN;
 4:     if $r$ is a server node then
 5:         Allocate $N \times N_{ij}/R$ VMs into root $r$;
 6:         The rest $N - N \times N_{ij}/R$ VMs are allocated based on the
            proportion of virtual load at each branch of the upper
            edge switch of root $r$ iteratively;
 7:     else if $r$ is a switch node then
 8:         Allocate the computation demand $N$ based on the pro-
            portion of virtual load at each branch from the root $r$
            iteratively;
 9:     Communication demands of placed VMs are split into
        paths based on the previous fixed flow schedule rule;
10:     Update the value of $R$ according to the distribution of
        existing loads;
11: else if $N > R$ then
12:     Reject;
```

3.2 Properties

This subsection studies several properties of the optimal solution for the fat tree DCN.

Theorem 6. *The optimal solution determines the MAL for a given fat tree under the hose model.*

Proof. Based on the fat tree topology, our proof starts at the bottom layer and progresses toward the core layer. In the bottom layer, we consider $m/2$ PMs connecting one edge switch. We first prove that $\sum_{j=1}^{j=m/2}\min\{N_{ij}, L_{ij}\}$ is the maximum virtual load by contradiction. We suppose that extra VMs can be placed in this virtual load, and we call the extra part λ. At least a portion of a branch exists that can accommodate λ, which is denoted by τ. The load on that branch is λ plus $\min\{N_{\tau j}, L_{\tau j}\}$, which overloads either the computation or the communication capacity. At the edge switch layer, we again consider $m/2$ branches connecting the aggregation switches $\sum_{j=m}^{j=2m-1}\min\{V_{ij}, L_{ij}\}$. We apply the same argument from the bottom layer to the edge layer, and the overload occurs on the bottlenecks (PMs or links) of the virtual load. The argument for the edge layer still can also be applied to upper layers. □

Theorem 7. *Given a fat tree that has a fixed flow schedule rule, the PPCC can obtain the maximum elasticity.*

Proof. Based on Theorem 1, we know that the maximum load in one layer is determined by the load proportion of branches. Since the flow schedule rule is fixed, the communication demands are related to the distribution of loads. If there is any deviation from the proportion that reduces the growth rate of other branches, the elasticity will reduce. Therefore, the PPCC obtains maximum elasticity for the fat tree that has a fixed flow schedule rule. □

4 Experiments

This section discusses the experiments based on our testbed, and presents various extensive simulations. All the results are shown from different perspectives to validate the effectiveness of the proposed policies.

4.1 Evaluations on Testbed

4.1.1 Testbed Configuration

Our testbed is constructed by our lab, and the topology is shown in Fig.9, which contains two Cisco switches (eight ports) and three Pica switches (48 ports). Each Pica switch connects two 64 bits Dell Power Edge R210 servers, and each server has 2.4 GHz CPU and 4 GB memory. All servers are accessible via the connections offered by the switch connections, and the capacities of the physical links are 1 Gbps. Each Pica switch connects two 64 bits Dell Power Edge R210 servers, and each server has 2.4 GHz CPU and 4 GB memory. Grnlntrn is a controller connected to port 10 on the Cisco switch, which is constructed by a Dell Power Edge R210 server. All servers are still accessible via the connections offered by the Cisco switch connections in Fig.9, and the capacities of the physical links are the same: 1 Gbps.

4.1.2 Evaluations for the Tree DCN

In this subsection, we focus on the capacity constraints which come from the computation and communication resource in the DCN. We evaluate the performance of the DNC by focusing on the transmission time for files. For each link in the testbed, we use bandwidth control on the physical ports in the switches Pica8-1, Pica8-2, and Pica8-3 to create a heterogeneous DCN topology. According to the example in Fig.7, we set the capacities of the links between switches and servers to be 0.7 Gbps, 0.4 Gbps, 0.6 Gbps, and 0.5 Gbps. The upper two links that connect two pairs of switches (Pica8-1 and Pica8-2, Pica8-1 and Pica8-3) are 1 Gbps and 0.6 Gbps respectively. Testbed validation is conducted in two settings. The first one focuses on performance

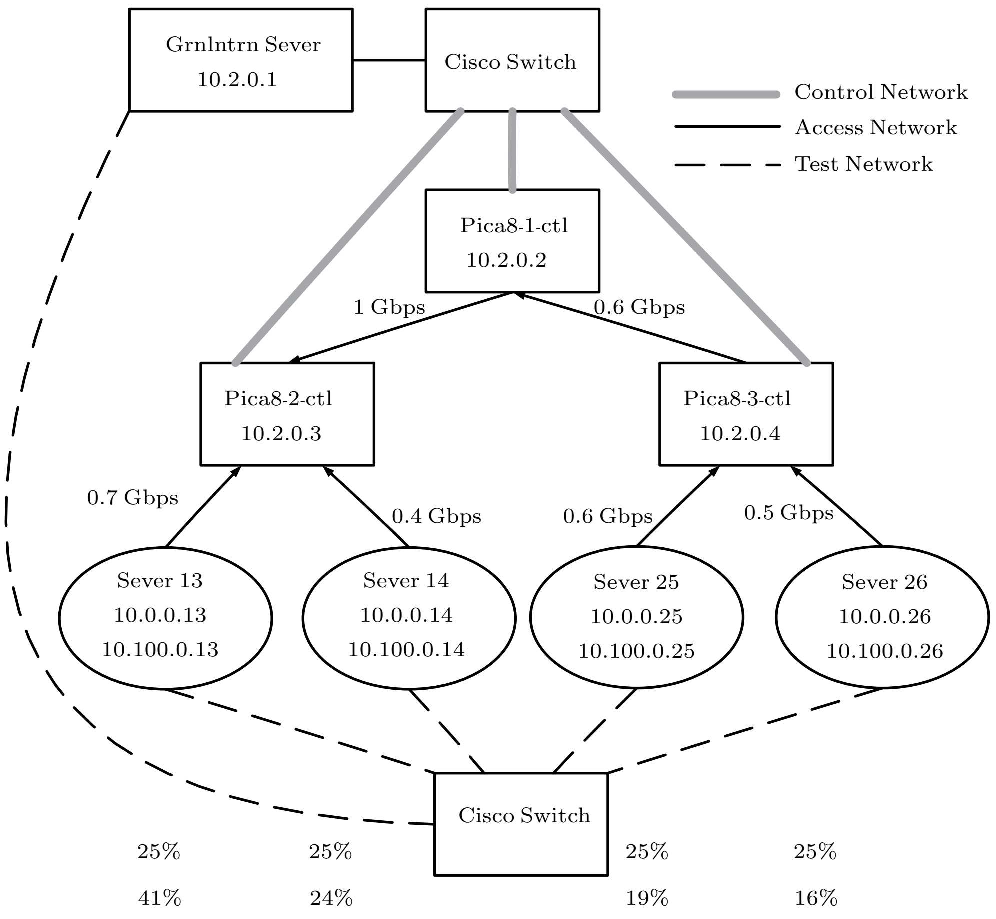


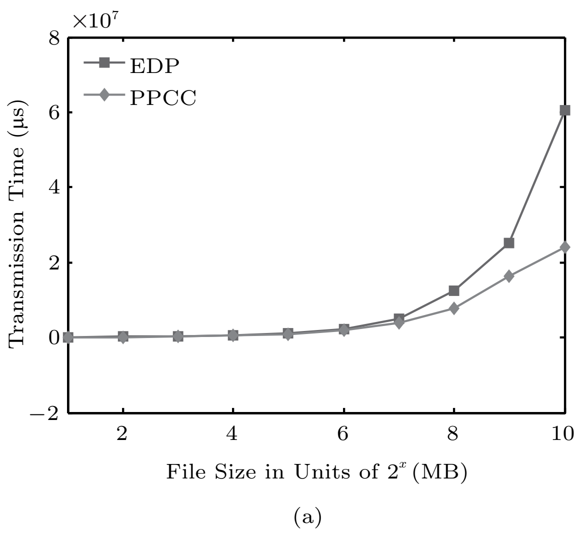
Fig.9. Topology of the testbed.

comparison between maximum elastic scheduling and equally distributed placement. As shown in Fig.9, each of four nodes (servers) receives 25% of load share based on equal partition. Under elastic scheduling, the optimal load distribution among four nodes from left to right is 41%, 24%, 19%, and 16%, respectively, which is based on the link bandwidth distribution as shown in Fig.9. To test scheduling elasticity, we conduct broadcasting applications (i.e., one node sends a message to all the other nodes), where the amount of broadcasting is based on the load share of each node. We increase the size of broadcast message (called file size) to test the scheduling elasticity in terms of communication delay. The second test is focused on how the efficiency of ECMP deteriorates when link bandwidth is no longer homogeneous at the same layer. We trace the transmission time of the scaling files between server 13 and servers 14, 25, and 26 respectively. Let each server be the sender, and the others be the receivers. The size of the file fluctuates from 2 MB to 1 GB, and the trace result appears in Fig.10. Equally distributed placement (EDP) scheme is used for comparison with our proportion with physical combinational capacities (PPCC) scheme. We analyze the influence on both computation and communication resource constraints, and we have two main observations. 1) Either computation or communication constraint may become the bottleneck of the maximum elasticity. As the setting of our evaluation for the tree DCN, each PM holds one VM with 2 GB memory resource allocation. The maximum elasticity on PM is 100%. However, the maximum elasticity on PL is only 20% of EDP and 50% of PPCC, respectively. 2) The transmission time of users under different schemes is significantly different. As shown in Fig.10(a), the transmission time for the file increases with the scaling of the file size gradually. When the file size is under the capacity of the physical links, the performances of EDP and PPCC are nearly the same. If we scale the size of file without considering the link capacity, the performance of EDP decreases significantly, as shown in Fig.10(b).

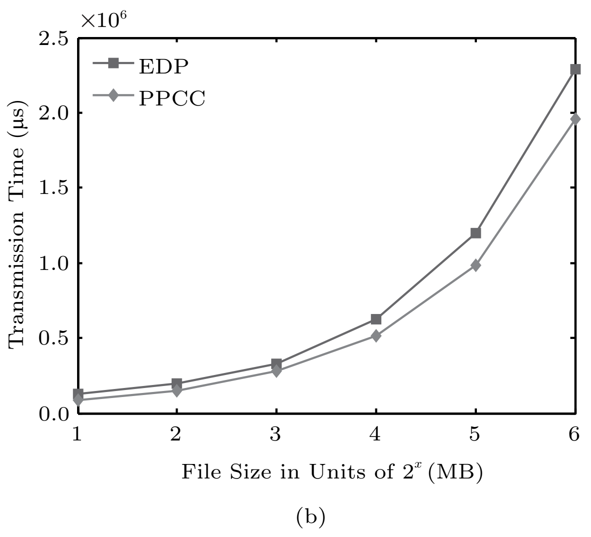
4.1.3 Evaluations for Fat Tree DCN

In this subsection, we add three virtual switches on each SDN switch (Pica8-1, Pica8-2, and Pica8-3) to simulate one pod of the fat tree with $k = 4$, as shown in Fig.11. For each switch we have three available physical ports P_1, P_2, and P_3. We divide the port P_1 into two virtual ports for each switch of Pica-2 and Pica-3. We also set server 13 as the sender, and servers 14, 25, and 26 as the receivers. The capacity of the physical link between Pica8-1 and Pica8-2 is 1 Gbps. We use bandwidth control to limit the flows passing through each port. Here, we set an adjustment factor δ, where $0 \leqslant \delta \leqslant 1$. The capacity of virtual link through port P_1 is δ Gbps, and another port P_2 is $(1-\delta)$ Gbps. Considering the constraints on the computation and communication, the size of the files fluctuates from 2 MB to 1 GB, and we set $\delta = 0.2$.

Based on the trace result that appears in Fig.12, we have the following observations. In our evaluations, we consider performances of two protocols which are equal cost multiple path (ECMP) and Non-ECMP. When the



(a)



(b)

Fig.10. Transmission time for tree DCN.

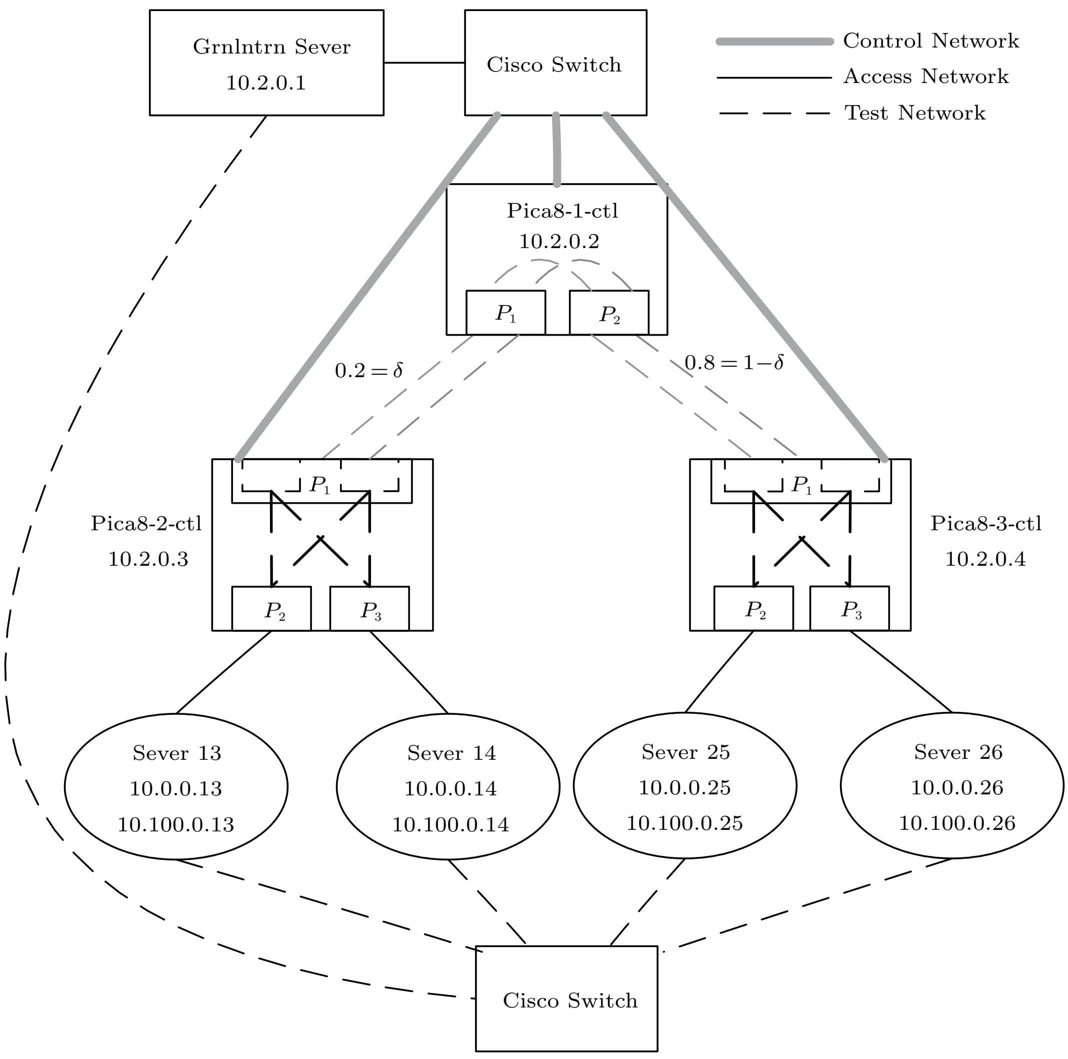


Fig.11. Transmission time for fat tree DCN.

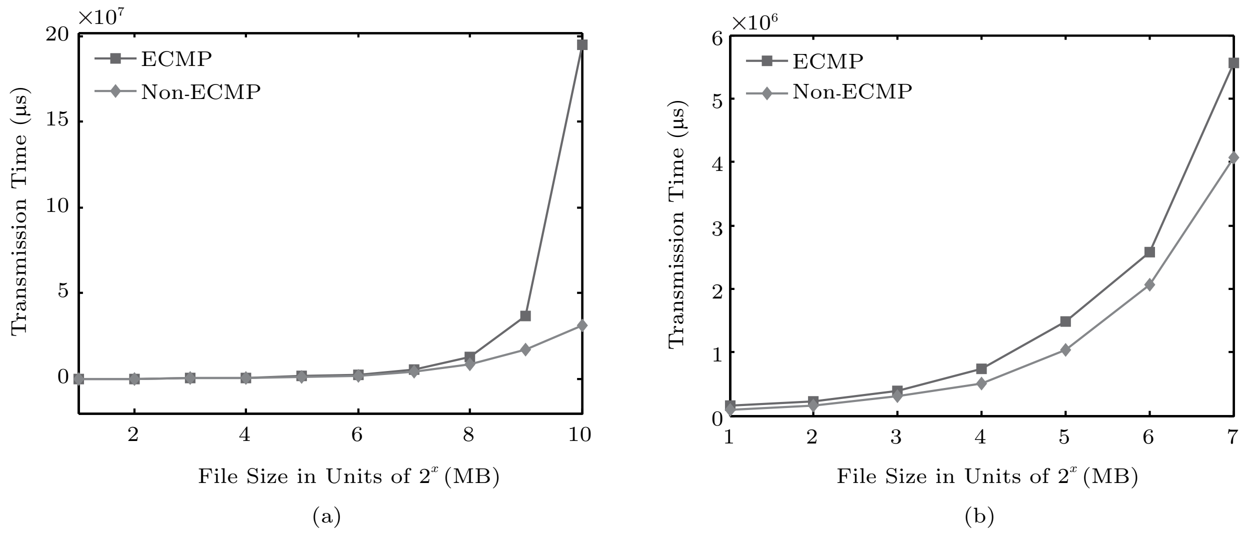


Fig.12. Transmission time for fat tree DCN.

file size is under 0.5 GB, there is no bandwidth constraint on any virtual link. As shown in Fig.12(b), although the transmission time under the Non-ECMP is shorter than that under ECMP, the gap is only nearly 6.5%. When the file size scales larger than 0.5 GB, one virtual link with 0.2 GB capacity will be the bottleneck. As shown in Fig.12(a), the transmission time increases sharply. Combining these two groups of evaluations on tree and fat tree DCNs, we can see that it is possible to accurately calculate the MAL for the DCN before doing the placement by considering both the computation and the communication constraints. This can effectively improve the flexibility of the data center and reduce the transmission delay. According to the observations of tree and fat tree DCNs, more simulations are conducted in Subsection 4.2.

4.2 Simulation Comparisons

In this subsection, we conduct various simulations on tree and fat tree DCNs.

4.2.1 Simulation Comparisons for Tree DCN

Basic Setting. Due to the limited size of our testbed, we do more simulations on larger-scaled DCNs to illustrate the effectiveness of our solutions. The DCNs are modeled as strict binary trees with different levels, where $k = 4$, $k = 5$, and $k = 6$. The number of nodes (i.e., PMs) ranges from 5 to 30. The node space (i.e., PM capacity) is heterogeneous and ranges from 0 to 100 units. The unit of the resource is slotted, which can be easily interpreted to be 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 of the average node space to the 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. 1) Equally distributed placement (EDP): the VMs are evenly assigned into the nodes in the tree. 2) Proportion with physical machine capacities (PPMC): the VMs are assigned into the nodes proportional to the space. 3) Proportion with physical link capacities (PPLC): the VMs are assigned into the nodes proportional to 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 Fig.13, 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 PMs' capacities and the physical links' capacities. The center point gradually shifts downward from the root with the bandwidth limitation. Fig.14 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. 1) When the ratio of the PMs' capacities to 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 the optimal solutions, as shown in Fig.14. 2) When the size of the tree is scaling, the upper links may be the bottleneck for communication between PMs. The locality of the maximal MAL may be shifted downward from the root, and the gap between the simple and the 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 Fig.13 and Fig.14(c).

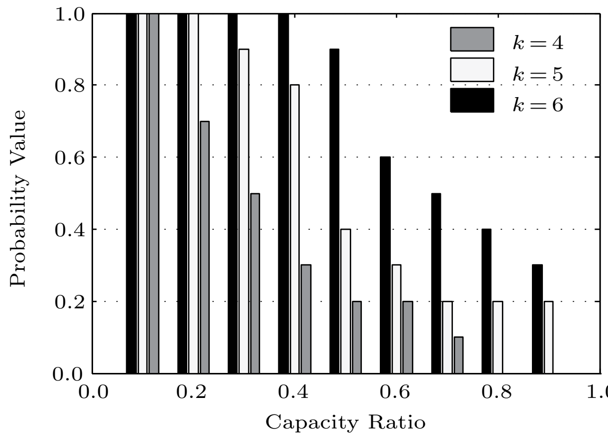


Fig.13. Probability that the maximal MAL exists at the root in tree DCN.

Analysis of Elasticity with Placement Under the Maximal MAL. We place VMs based on the proportion with physical combinational capacities (PPCC). Each virtual request is iteratively placed using the proportion of the bottleneck resource, which may be either PM capacities or physical link capacities. The value of the resource demand for each request lies within $[0, 200]$. For each group, we do the placement based on the available resource and calculate the elasticities. We average the

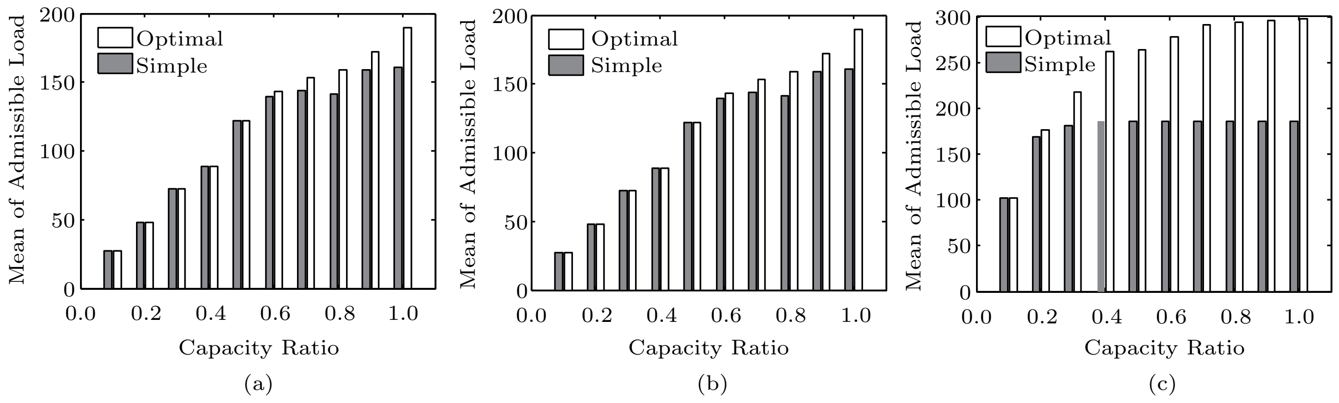


Fig.14. Comparison of the elasticities for simple and optimal solutions. (a) $k = 4$. (b) $k = 5$. (c) $k = 6$.

results of the placement 10 times for virtual requests with different algorithms. We have the following observations. 1) As shown in Figs.15(a)–15(c), the elasticity grows with the scaling of the tree size. 2) The elasticity depends on the capacity ratio. As the blue line shows in Fig.15, the combinational elasticities under the strategies decrease with the increasing ratio. We can have that when the ratio of the average node space is lower than the average link bandwidth, the bandwidth is not the bottleneck. 3) The elasticity also depends on the placement strategy. As shown in the green lines in Fig.15, 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 with the baseline algorithms. It improves the resultant elasticity by 16.2%, 17.2%, and 11.9% under $k = 4$, 5, and 6, respectively.

4.2.2 *Simulation Comparisons for Fat Tree DCN*

Basic Setting. The DCN is modeled as a fat tree with the number of ports $m = 4$, $m = 6$, $m = 8$, $m = 10$, and $m = 12$. The number of corresponding PMs ranges from 16 to 432. The capacity of each PM is heterogeneous and ranges from 0 to 10 at the bottom layer. The unit of the resource is slot, which can be easily interpreted to a real configuration. Each slot can hold on one VM, and the bandwidth demand between per-pair of VMs is 1 Gpbs. There are $\frac{k}{2}$ edge switches connected to an aggregation switch, and the same amount of aggregation switches are connected to the core switch. Our evaluation is divided into two groups according to the configuration of the fat tree DCN. One evaluation group is the semi-homogeneous configuration in which the capacities of physical links in one layer are the same, but the capacities of PMs are different. For the semi-homogeneous scenario, we vary the capacities of physical links for each layer, from the bottom up as 5 Gbps, 10 Gbps, and 20 Gbps, respec-

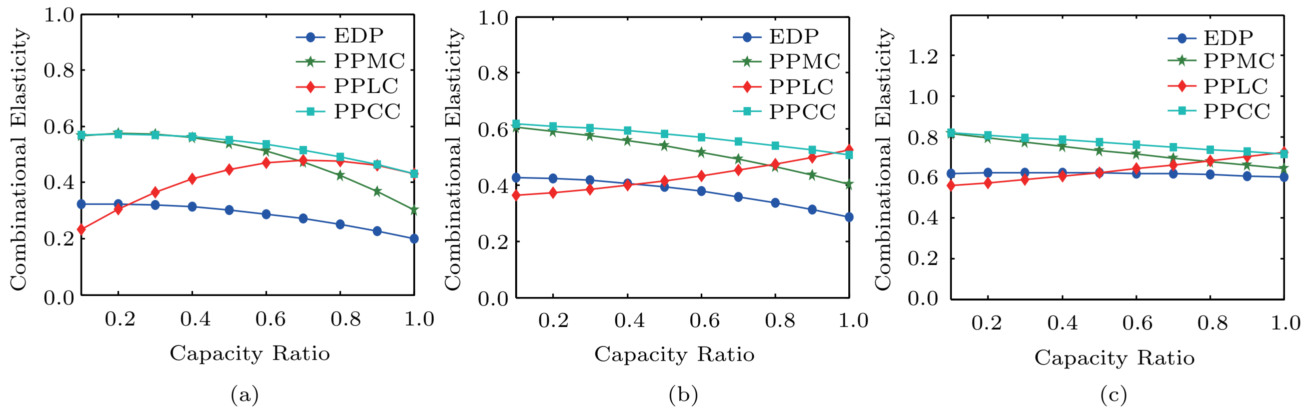


Fig.15. Comparison of the elasticities for baseline and optimal solutions. (a) $k = 4$. (b) $k = 5$. (c) $k = 6$.

tively. Another evaluation is conducted for the heterogeneous configuration. Setting of the PMs capacities is the same in the heterogeneous configuration as in the semi-homogeneous configuration. We set the link capacity ranges as $[0, 10]$ Gbps, $[0, 15]$ Gbps, and $[0, 30]$ Gbps in each layer, and the amount of VMs in each set is evenly distributed between 0 and 100.

Simulation Results. We first analyze the maximal value of the MALs under both the semi-homogeneous and heterogeneous cases. Fig.16(a) shows the root distribution of the different scales of fat tree DCNs, when the number of the switches' ports is $m = 4$, $m = 6$, $m = 8$, $m = 10$, and $m = 12$, respectively. For each fat tree DCN, we generate the fat tree DCNs with various capacities for 10 times, and we calculate the value and point out the location of the maximal MALs.

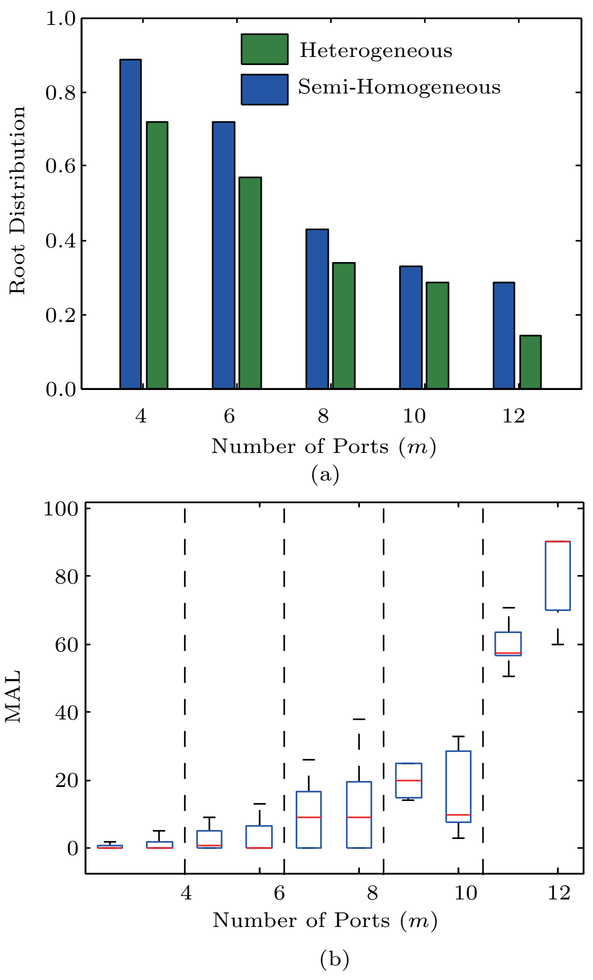


Fig.16. (a) Root distribution of the different scales of fat tree DCNs. (b) MALs for fat trees.

We have the following observations based on the simulation results. 1) As shown in Fig.16(a), for each group experiment, the probability that the maximal MAL exists on the core layer decreases with the scaling capacity of the fat tree DCNs in both the semi-homogeneous and heterogeneous cases. This means that with the scaling for fat tree DCNs, the bandwidth of upper links becomes the main limitation for communication between PMs. Fig.16(b) shows a comparison of the MALs between the core switch aggregation and the root r. Two solutions calculate the MALs under fat tree DCNs with various capacities, where $m = 4$, $m = 6$, $m = 8$, $m = 10$, and $m = 12$, respectively. The differences between these two solutions increase as the number of ports increases. Since the orientation of the aggregation depends on the order in which the virtual loads are received, and since the upper links have limited bandwidth, the root gradually shifts down from the core layer. 2) The localities of the maximal MALs may be different for different configurations in the fat tree DCNs. As shown in Fig.16(a), the probability of the blue columns is higher than that of the green ones. In the semi-homogeneous case, the capacities of the physical links in a layer are the same, and thus the single link between the loads (PMs or virtual loads) and the switches cannot be the bottleneck. However, they can be the bottleneck in the heterogeneous case because the link capacities are determined in a range. Therefore, in the heterogeneous case, the probability that the maximal MALs exist in the core layers is lower than that in the semi-homogeneous case. Next, we start to analyze the elasticity for placement under the maximal MAL. In order to simplify our simulation, we use a virtual cluster to denote a set of VMs. We compare our provisioning algorithm with the same three baseline algorithms, EDP, PPMC, and PPLC.

Fig.17 presents the elasticity of VMs provision under the fat tree DCNs. The group is divided by the number of the switches' ports: $m = 4$, $m = 6$, $m = 8$, $m = 10$, and $m = 12$, respectively. For each group, we use the same four algorithms: EDP, PPMC, PPLC, and PPCC. We calculate an average of 10 times of the elasticity for various virtual clusters that have the amount of VMs evenly distributed between 0 and 100. Additionally, we have the following observations. 1) The fluctuation of the elasticity depends on the scales of the virtual clusters. As shown in Fig.17, the elasticities of all the algorithms decrease with the increase in the amount of VMs in the virtual cluster. 2) The elasticity for the virtual cluster depends on the provisioning algorithms. The performance of each algorithm depends on the provisioning of the virtual cluster. For each group,

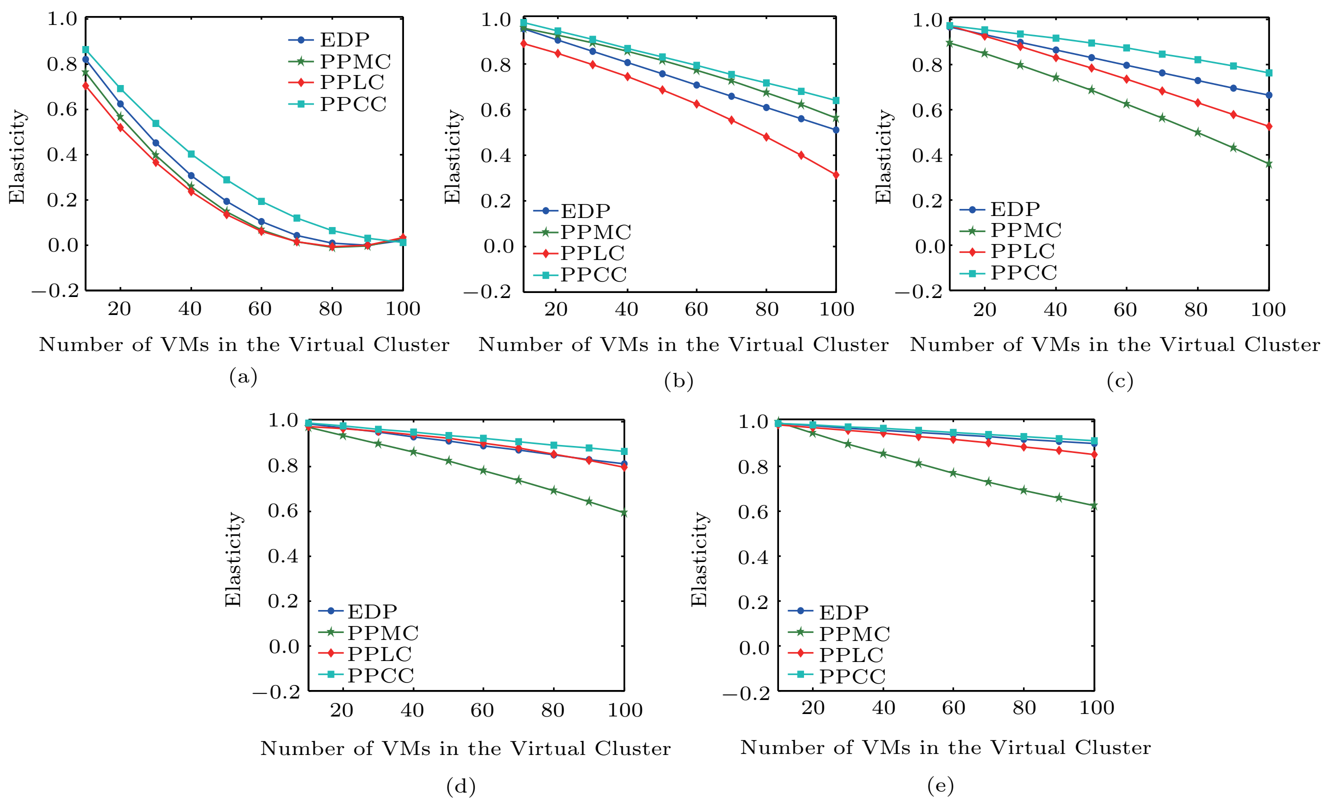


Fig.17. Elasticities for the fat trees. (a) $m = 4$. (b) $m = 6$. (c) $m = 8$. (d) $m = 10$. (e) $m = 12$.

we can see that the performance of the greedy algorithm decreases significantly with the increase in the amount of VMs in the virtual cluster. The performance of the PPMC and the PPLC algorithms improve with the scaling of the fat tree DCNs. However, they all have lower elasticities when the scales are small, i.e., $m = 4$. The performance of EDP depends on the capacities of the PMs. If the distribution of the capacities of the PMs is uneven, the performance will decrease significantly. Compared with PPMC, PPLC, and EDP, PPCC has the best performance in the elasticity across the various fat tree DCNs. The gap is more obvious with the increase in the scale of the fat tree DCN. We can see that the elasticity between different algorithms for the virtual cluster is much lower with $m = 4$ than with $m = 12$ in the fat tree DCNs.

5 Related Work

Virtualization technology ensures application, isolation, and at the same time allows for the utilization of the PM. Much work has been done in the VM placement in cloud-based DCNs with constraints that include power[2,10] and performance interference[11,12], reliability[13,14], energy consumption[15], traffic variability[16,17], and traffic minimization[18]. [19] also discusses some other practical factors in VM scheduling.

There are two common models for DCN virtualization which are the pipe model and the hose model. In the pipe model, the customer needs to know the complete traffic load between each pair of VPN endpoints[8]. However, in the hose model, the service provider supplies the customer with certain guarantees for the traffic that each endpoint sends to and receives from other endpoints of the same VPN[4,20]. One extension of the hose model is the virtual oversubscribed cluster, which guarantees a specified minimal bandwidth between tenants' virtual machines based on an over-subscription factor[21]. Another combination abstraction of the hose model and the pipe model is tenant application graph[22], which preserves inter-component structure as the pipe model in each tier and aggregates pipes as the hose model between a pair of communicating tiers. Among many hose model based applications, Kumar *et al.*[20] connected VPN endpoints using a tree structure and aimed to optimize the total bandwidth reserved on edges of the VPN tree under single path routing. Erlebach and Maurice[23] did an extension which presents an optimal polynomial-time algorithm for multi-path routing. The hose model is also applied into the virtual cluster placement. Ballani *et al.*[21] and Zhu *et al.*[24] aimed to find the smallest subtree using a greedy algorithm that can fit under both the homogeneous and the heterogeneous bandwidth demands separately. To optimize resource usage, Dutta *et al.*[25] presented a dynamic programming algorithm for computing the optimal embedding with minimum congestion.

Elasticity has been considered one of the central attributes when estimation cannot be easily obtained[26,27]. 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[28]. In order to define a measure of the elasticity, Shawky and Ahmed[29] provided a set of benchmarks for cloud computing performance. In [3, 30], the authors considered the elasticity-aware VM placement problem in 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 bandwidths of the links at the same level are the same, but the ones at different levels are different. The scheme proposed in this paper extends the optimality to general tree structures.

6 Conclusions

This paper proposed 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 and communication elasticities. 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. We first considered the tree-structured DCN and offered a distributed, optimal solution that computes the maximum admissible load and performs the maximum elastic scheduling of any admissible load. Based on the assumptions and conclusions, we extended it to the multiple paths case with a fat tree DCN, and we discussed the optimal solution for computing the MAL with both computation and communication constraints. We presented the provisioning scheme with the maximum elasticity for the VMs, which comes with provable optimality guarantee for a fixed flow scheduling strategy in a fat tree DCN. Compared with the existing baseline algorithms, the simulation results showed that our algorithm can improve the resultant elasticities by 15.1% and 21.8% in the binary tree and the fat tree DCNs, respectively. Moreover, the evaluation results on the real testbed validated that our algorithm can improve the elasticity of the DCN and reduce the transmission delay.

References

[1] Bari M F, Boutaba R, Esteves R *et al.* Data center network virtualization: A survey. *IEEE Communications Surveys and Tutorials*, 2013, 15(2): 909-928.

[2] Mann Z A. Allocation of virtual machines in cloud data centers — A survey of problem models and optimization algorithms. *ACM Computing Surveys*, 2015, 48(1): Article No. 11.

[3] Li K K, Wu J, Adam B. Elasticity-aware virtual machine placement for cloud datacenters. In *Proc. the 2nd International Conference on Cloud Networking*, November 2013, pp.99-107.

[4] Duffield N G, Goyal P, Greenberg A *et al.* A flexible model for resource management in virtual private networks. In *Proc. the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, August 1999, pp.95-108.

[5] Lee Y T, Sidford A, Wong S C W. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proc. the 56th Annual Symposium on Foundations of Computer Science*, October 2015, pp.1049-1065.

[6] Leiserson C E. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE transactions on Computers*, 1985, C-34(10): 892-901.

[7] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In *Proc. the ACM SIGCOMM 2008 Conference on Data communication*, August 2008, pp.63-74.

[8] Davie B S, Rekhter Y. MPLS: Technology and Applications (1st edition). Morgan Kaufmann, 2000.

[9] Liu Y, Muppala J K, Veeraraghavan M *et al.* Data Center Networks: Topologies, Architectures and Fault-Tolerance Characteristics. Springer International Publishing, 2013.

[10] Wang R X, Wickboldt J A, Esteves R P *et al.* Using empirical estimates of effective bandwidth in network-aware placement of virtual machines in data centers. *IEEE Transactions on Network and Service Management*, 2016, 13(2): 267-280.

[11] Kusic D, Kephart J O, Hanson J E *et al.* Power and performance management of virtualized computing environments via look a head control. *Cluster Computing*, 2009, 12(1): 1-15.

[12] Xu F, Liu F, Liu L *et al.* iAware: Making live migration of virtual machines interference-aware in the cloud. *IEEE Transactions on Computers*, 2014, 63(12): 3012-3025.

[13] Yang S, Wieder P, Yahyapour R *et al.* Reliable virtual machine placement and routing in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 2017, 28(10): 2965-2978.

[14] Guo C X, Lu G H, Wang H J *et al.* SecondNet: A data center network virtualization architecture with bandwidth guarantees. In *Proc. the 6th International Conference*, November 2010, Article No. 15.

[15] Deng W, Liu F, Jin H *et al.* Reliability-aware server consolidation for balancing energy-lifetime tradeoff in virtualized cloud datacenters. *International Journal of Communication Systems*, 2014, 27(4): 623-642.

[16] Liu F M, Guo J, Huang X *et al.* eBA: Efficient bandwidth guarantee under traffic variability in datacenters. *IEEE/ACM Transactions on Networking*, 2017, 25(1): 506-519.

[17] Li X, Wu J, Tang S *et al.* Let's stay together: Towards traffic aware virtual machine placement in data centers. In *Proc. the 33rd IEEE International Conference on Computer Communications*, April 2014, pp.1842-1850.

[18] Meng X Q, Pappas V, Zhang L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proc. the 29th IEEE International Conference on Computer Communications*, March 2010, pp.1154-1162.

[19] Xu F, Liu F M, Jin H *et al.* Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE*, 2014, 102(1): 11-31.

[20] Kumar A, Rastogi R, Silberschatz A *et al.* Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Transactions on Networking*, 2002, 10(4): 565-578.

[21] Ballani H, Costa P, Karagiannis T *et al.* Towards predictable datacenter networks. In *Proc. the 2011 ACM Conference on SIGCOMM*, August 2011, pp.242-253.

[22] Lee J, Turner Y, Lee M *et al.* Application-driven bandwidth guarantees in datacenters. In *Proc. the 2014 ACM Conference on SIGCOMM*, August 2014, pp.467-478.

[23] Erlebach T, Ruegg M. Optimal bandwidth reservation in hose-model VPNs with multi-path routing. In *Proc. IEEE INFOCOM 2004*, March 2004, pp.2275-2282.

[24] Zhu J, Li D, Wu J P *et al.* Towards bandwidth guarantee in multi-tenancy cloud computing networks. In *Proc. the 20th IEEE International Conference on Network Protocols*, October 2012.

[25] Dutta D, Kapralov M, Post I *et al.* Optimal bandwidth aware VM allocation for Infrastructure-as-a-Service. arXiv:1202.3683, 2012. https://arxiv.org/abs/1202.3683, Feb. 2018.

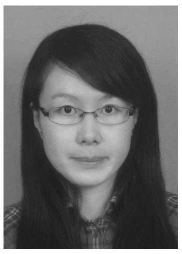
[26] Plummer D C, Smith D M, Bittman T J *et al.* Five refining attributes of public and private cloud computing. Technical Report, Gartner, 2009. http://www.gartner.com/Display-Document, May 2018.

[27] Lu S B, Fang Z Y, Wu J *et al.* Elastic scaling of virtual clusters in cloud data center networks. In *Proc. the 36th International Performance Computing and Communications Conference*, December 2017, pp.1-8.

[28] Herbst N R, Kounev S, Reussner R. Elasticity in cloud computing: What it is, and what it is not. In *Proc. the 10th International Conference on Autonomic Computing*, June 2013, pp.23-27.

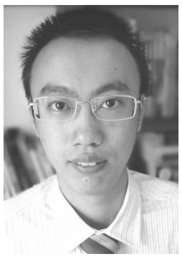
[29] Shawky D M, Ali A. Defining a measure of cloud computing elasticity. In *Proc. the 1st International Conference on Systems and Computer Science*, August 2012, pp.1-5.

[30] Li K K, Wu J, Blaisse A. Elasticity-aware virtual machine placement in k-ary cloud data centers. *Parallel and Cloud Computing*, 2014, 3(2): 22-31.

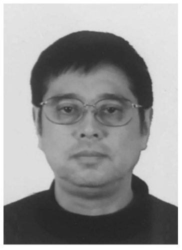


Shuai-Bing Lu is currently a Ph.D. candidate in computer science and technology of Jilin University, Changchun. She is supported by the China Scholarship Council as a visiting scholar supervised by Prof. Jie Wu in the Department of Computer and Information Sciences at Temple University (2016–2018), Philadelphia. She is a student member of IEEE. Her current research focuses on distributed computing, cloud computing and fog computing.

Jie Wu is the director of the Center for Networked Computing and Laura H. Carnell professor at Temple University, Philadelphia. He also serves as the director of International Affairs at College of Science and Technology. He served as the chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation of USA and was a distinguished professor at Florida Atlantic University, Boca Raton. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and Journal of Parallel and Distributed Computing. Dr. Wu was the general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, and program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a fellow of IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.



Huan-Yang Zheng received his B.Eng. degree in telecommunication engineering from Beijing University of Posts and Telecommunications, Beijing, in 2012. He is currently a Ph.D. candidate in the Department of Computer and Information Sciences, Temple University, Philadelphia. His current research focuses on mobile networks, social networks, and cloud systems.



Zhi-Yi Fang received his Ph.D. degree in computer science and technology from Jilin University, Changchun, in 1998. He currently works there as a professor in the College of Computer Science and Technology of Jilin University, Changchun. He was a senior visiting scholar at the University of Queensland, Brisbane, from 1995 to 1996, and at the University of California, Santa Barbara, from 2000 to 2001. He is a member of the China Software Industry Association (CSIA) and a member of the Open System Committee of China Computer Federation (CCF). His research interests include distributed/parallel computing systems, mobile communication, and wireless networks.