Received: 20 February 2017 | Revised: 3 September 2017 | Accepted: 12 October 2017

DOI: 10.1002/cpe.4383

WILEY

RESEARCH ARTICLE

On efficient virtual cluster scaling across geo-distributed data centers

Xinping Xu | Wenxin Li | Heng Qi | Keqiu Li

School of Computer Science and Technology, Dalian University of Technology, Dalian, China

Correspondence
Heng Qi, Keqiu Li, Dalian University of Technology, Dalian 116023, China.
Email: hengqi@dlut.edu.cn; keqiu@dlut.edu.cn

Funding information
National Key Research and Development Program of China, Grant/Award Number: 2016YFB1000205; State Key Program of National Natural Science of China, Grant/Award Number: 61432002; NSFC, Grant/Award Number: 61772112, 61272417, 61300189, 61370199 and 61672379.

Summary

Virtual cluster has recently emerged as a common abstraction for cloud applications or tenants to specify and reserve resources. Such virtual cluster brings valuable insights into the cloud elasticity when scaling up or down the number of resources on demand. Unfortunately, for global-scale applications running on geo-distributed datacenters, it is always a challenge to scale the virtual cluster. Due to the fact that the inter-datacenter bandwidth is an expensive and scarce resource, it is increasingly important yet typically hard to achieve cost-minimizing bandwidth guarantees when scaling. However, existing approaches mainly focus on the scaling within intra-datacenter networks and cannot be simply extended to the inter-datacenter scenario. In this paper, we study the problem of scaling up a virtual cluster with consideration of both bandwidth cost minimization and bandwidth guarantees fulfillment targeting inter-datacenter networks. Specifically, we first propose an efficient algorithm to scale up the virtual cluster without changing its original VM placement. With the observation that such VM placement can hinder the cluster scalability, we further present an optimized algorithm, which exploits VM migration when scaling. Finally, we conduct extensive simulations to demonstrate the effectiveness of our algorithms, in terms of both bandwidth cost and the acceptance rate of scaling requests with bandwidth guarantees.

KEYWORDS

geo-distributed data center, resource scaling, virtual cluster

1 | INTRODUCTION

A key feature of cloud computing is its elasticity and capability of scaling up and down the resources on demand, for cloud applications to adapt to the changes in workload.[1] In the meanwhile, the virtual cluster has recently emerged as a common abstraction for cloud applications or tenants to specify their resource demands.[2-5] This eventually leads to an important problem of virtual cluster scaling, which is crucial to the performance of cloud applications.

While recognizing the significance of virtual cluster scaling to cloud applications, we observe that a number of such applications are deployed on a geographically distributed infrastructure,[6-8] ie, datacenters located in different regions across the globe. Notable examples include Web search,[6,9] video streaming,[10,11] social networking,[12] and wide area big data analytics.[13] The trend towards geographical deployment of applications will only continue and even increasingly include smaller enterprises, with the success of cloud computing. This evolution of application deployment motivates us to think about a question: *How to scale up a virtual cluster across geo-distributed datacenters?*

To scale up a virtual cluster across multiple datacenters, we argue that the following two basic requirements should be jointly considered. *First*, cloud applications, from their individual perspectives, have a strong desire for predictable network performance. Based on a state-of-the-art study, bandwidth guarantee is an efficient approach to achieve the predictable performance.[14] Because of offering a strong isolation for applications, it can even provide predictable job completion time for those network-intensive applications (eg, MapReduce across geo-distributed datacenters). *Second*, deploying virtual clusters across geo-distributed datacenters consumes significant amount of inter-datacenter bandwidth, which, however, is often the scarcest, most volatile, and/or most expensive resource. In particular, the annual bandwidth cost may be up to hundreds of millions of dollars.[9] Therefore, *from an economic perspective*, the bandwidth cost should be considered as a fundamental factor when scaling the virtual cluster across multiple datacenters.

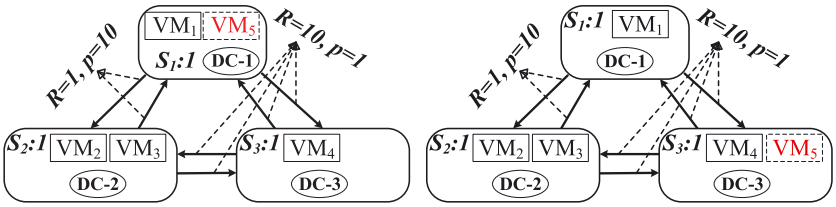


FIGURE 1 An illustrative example of scaling up a virtual cluster from $< 4, 1 >$ to $< 5, 1 >$ in a 3-datacenter setup where 4 or 5 is the number of VMs and 1 is the bandwidth requirement between each VM-pair. Here, *S* represents the number of empty slots of each datacenter, *R* denotes the residual bandwidth, and *p* stands for the price per unit bandwidth, of each inter-datacenter link, respectively

To the best of our knowledge, existing methods on virtual cluster scaling are only applicable to intra-datacenter networks. Most of them focus on dynamically adjusting the number of VMs in a virtual cluster, with ignoring bandwidth guarantees when scaling.[15-17] A recent study in Yu and Cai[18] addresses the bandwidth guarantee problem along with the virtual cluster scaling. The scaling method in Yu and Cai[18] actually works within the intra-datacenter network with a tree-like topology. It first searches empty slots from the leave nodes and then traverses the tree topology level-by-level in a bottom-up manner till the scaling succeeds. However, it cannot be simply extended to inter-datacenter networks.

To have a comprehensive understanding, we present an illustrative example in Figure 1, where a virtual cluster is desired to be scaled from $< 4, 1 >$ to $< 5, 1 >$ in a 3-datacenter setup. Each datacenter only has 1 empty slot. Note that there could be significant *heterogeneity* both in the bandwidth capacity and in the price per unit bandwidth of different inter-datacenter links.[6,9,19] A key limitation of Yu and Cai[18] is that such *heterogeneity* is not perceived. Suppose the scaling method of Yu and Cai[18] is being applied, followed by the bottom-up traversing manner, it may possibly place the new VM_5 into datacenter 1, as shown in Figure 1A. This is because that placing VM_5 into datacenter 1 does not overload any datacenter or oversubscribe any inter-datacenter link. As such, the corresponding bandwidth cost is 10 because one more unit of bandwidth has to be reserved on the link between datacenters 1 and 2, to guarantee the bandwidth for each VM-pair. On the contrary, one can easily check a better scaling solution is to place the new VM_5 into datacenter 3 instead, as illustrated in Figure 1B. In this scenario, the bandwidth cost can be significantly reduced to as minimum as 1; meanwhile, the guaranteed bandwidth requirement can also be satisfied for each VM-pair.

In this paper, we address the challenging problem of scaling up a virtual cluster across geo-distributed datacenters. Our primary goal is to minimize the inter-datacenter bandwidth cost and, at the same time, fulfill the bandwidth guarantee for each VM-pair within the cluster. Specifically, we consider two typical cases of this problem. *The first case* is to scale up a virtual cluster, without changing its original VM placement. To this end, we formulate an optimization problem, which takes account of both guaranteed bandwidth requirement and inter-datacenter bandwidth cost. To solve this optimization problem, we propose a dynamic programming algorithm, which searches for the placement of additional VMs at the lowest cost; meanwhile, it can still provide bandwidth guarantees for each pair of VMs. Unfortunately, such dynamic programming algorithm may not work in case that a virtual cluster cannot be scaled without changing its original VM placement. Hence, this emerges as *the second case* of this problem, which needs to leverage VM migration to scale up the virtual cluster while minimizing the sum of both bandwidth cost and migration cost. We also present an optimized algorithm to solve this case. Finally, we conduct extensive simulations to demonstrate the effectiveness of our algorithms. The results verify that our algorithms are capable of both reducing the inter-datacenter bandwidth cost and satisfying the guaranteed bandwidth requirement when scaling up a virtual cluster across geo-distributed datacenters.

The rest of this paper is organized as follows. Section 2 first studies the virtual cluster scaling problem in case the original VM placement cannot be changed. In Section 3, we focus on the virtual cluster scaling with VM migration enabled. In Section 4, we evaluate and analyze the performance of our proposed algorithms. In Section 5, we discuss current limitations and revelent future research. Section 6 summarizes the related work. Finally, conclusions are drawn in Section 7.

2 | VIRTUAL CLUSTER SCALING WITHOUT VM MIGRATION

In this section, we first describe the problem of scaling a virtual cluster across geo-distributed datacenters and then present a scaling algorithm in the case where the original VM placement cannot be changed.

2.1 | Virtual cluster scaling

We consider a virtual cluster, denoted as $< N, B >$, where N is the number of VMs and B is the bandwidth requirement between each VM-pair. In our analysis, we mainly focus on scaling a virtual cluster from $< N, B >$ to $< N', B >$ $(N' > N)$. To ease the presentation, the key parameters used throughout this paper are listed in Table 1. As we mentioned earlier, a number of cloud applications are increasingly deployed on a geographically distributed infrastructure. Hence, we consider that the virtual cluster is deployed across a set of datacenters. We suppose that every datacenter is connected to all other datacenters and accordingly use a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote the inter-datacenter network,

TABLE 1 Notations and definitions

Symbol	Definition
$\mathcal{V}$	The set of datacenters, $\mathcal{V} = \{1, 2, \dots, M\}$
$l_{i,j}, \mathcal{E}$	The set of inter-datacenter links, $l_{i,j} \in \mathcal{E}$
N, N'	The original cluster size (the number of VMs) and the new size ($N' > N$)
N_i	The number of VMs in the original cluster located in datacenter $i \in \mathcal{V}$, $\sum_i N_i = N$
N'_i	The number of VMs in the scaled cluster located in datacenter $i \in \mathcal{V}$, $\sum_i N'_i = N'$
$R_{i,j}$	The residual bandwidth of the inter-datacenter link, $l_{i,j} \in \mathcal{E}$
$p_{i,j}$	The price per unit bandwidth of the inter-datacenter link, $l_{i,j} \in \mathcal{E}$
S_i	The number of empty slots of datacenter i
B	The guaranteed bandwidth requirement of the virtual cluster
W	The cost of migrating one VM
C^{bw}	The total cost of inter-datacenter bandwidth reservations
C^{mgr}	The total cost of VM migrations
θ	The increment rate of the virtual cluster

where $\mathcal{V} = \{1, 2, \dots, M\}$ is the set of datacenters and $\mathcal{E}$ is the set of inter-datacenter links. In this case, let N_i stand for the number of VMs of the virtual cluster that are placed in datacenter $i \in \mathcal{V}$, similarly, S_i for the number of empty slots of datacenter i. It is clear that $\sum_i N_i = N$. For each inter-datacenter link $l_{i,j} \in \mathcal{E}$, let $R_{i,j}$ denote the residual bandwidth, and $p_{i,j}$ as the price per unit bandwidth.

Now, we are in a position to formulate the problem of scaling a virtual cluster with both bandwidth cost and bandwidth guarantees taken into account. Consider any inter-datacenter link $l_{i,j}$ that connects datacenter i and j. The virtual cluster has N_i VMs in datacenter i, and N_j in datacenter j. Then the maximum bandwidth required on link $l_{i,j}$ is $\min(N_i, N_j) * B$, because each VM cannot send or receive at a rate higher than B.[18] This implies that we need to reserve $\min(N_i, N_j) * B$ amount of bandwidth on each link $l_{i,j}$ to guarantee the bandwidth B for each VM-pair. When the virtual cluster $< N, B >$ is scaled up to $< N', B >$ without changing the original placement of N VMs, we have to additionally reserve $\Delta^{bw}_{i,j}$ amount of bandwidth on link $l_{i,j}$ accordingly. It is calculated as follows:

$$\Delta^{bw}_{i,j} = (\min(N'_i, N'_j) - \min(N_i, N_j)) * B, \tag{1}$$

where N'_i and N'_j are the number of VMs being placed in datacenter i and j, respectively, after the scaling. We now can formulate the cluster scaling problem **P1** of minimizing the bandwidth cost of inter-datacenter links while still guaranteeing the bandwidth requirement for each VM-pair, as follows:

$$\min_{N'_i} \quad C = \sum_{l_{i,j}} p_{i,j} \Delta^{bw}_{i,j} \tag{2}$$

$$\text{S.t.} \quad N'_i \geq N_i, \forall i, \tag{3}$$

$$\Delta^{bw}_{i,j} \leq R_{i,j}, \forall l_{i,j}, \tag{4}$$

$$N'_i - N_i \leq S_i, \forall i, \tag{5}$$

$$\sum_i N'_i = N'. \tag{6}$$

It is clear that the objective in Equation 2 represents the total bandwidth cost. Equation 3 means that in each datacenter, the number of VMs in the scaled cluster is always no less than that of the original, which also indirectly suggests that the VM placement for the original cluster is always not changed. Equation 4 enforces that the increase of bandwidth reservation must be no greater than the residual bandwidth of the corresponding inter-datacenter link. Similarly, Equation 5 implies that the increase of the number of VMs in each datacenter should be limited by its empty slots. Equation 6 essentially indicates that the sum of the number of VMs in each datacenter should be equal to N'.

2.2 | A scaling algorithm

We now present a dynamic programming algorithm to solve the virtual cluster scaling problem **P1**. Intuitively, it may be a step towards the right direction to leverage numerical methods (eg, simplex method[20]) to solve the problem **P1**. However, it usually requires numerous iterations and inherently increase the overall computation overhead of the algorithm. Furthermore, we may encounter a case where the decision variables computed are not integers, which is inconsistent with real-world situations. Hence, we are inspired to design an efficient yet lightweight cluster scaling algorithm instead, such that the overall computation overhead can be significantly reduced. Moreover, the algorithm can also be triggered in a *laissez-fair* manner on general cloud platforms. In other words, whenever a cloud application encounters a high workload and wants to scale up the size of its virtual cluster across geo-distributed datacenters, this scaling algorithm is triggered.

The key for scaling up a virtual cluster is summarized in Algorithm 1. The algorithm requires the information of inter-datacenter networks topology, the number of empty slots in each datacenter, the residual bandwidth, and the price per unit bandwidth of each inter-datacenter link, as well as the current VM placement for the virtual cluster to be scaled. Given these inputs, it generates a VM placement for the scaled cluster. To scale up a virtual cluster, the algorithm performs the following steps: it first identifies the increase in the size of the virtual cluster (Step 1), initializes some auxiliary variables (Step 2), then uses a **While** loop to scale the cluster to the target size in a gradual manner, ie, each time it scales the cluster with only one more VM. More precisely, as illustrated in Steps 5 to 13, it searches a datacenter that will produce the lowest bandwidth cost, also does not lead to overloading any associated links, as well as having sufficient empty slots to accommodate the new VM if placing one more. After finding such datacenter, the algorithm places the new VM in it (Step 14). Finally, for the datacenter being searched, the algorithm updates the number of empty slots and the residual bandwidth of all associated links (Steps 15-16).

It should be noted that this algorithm makes no changes to the original VM placement for the virtual cluster. In what follows, we present an optimized algorithm to cope with the case where VM migration is necessary and required.

3 | VIRTUAL CLUSTER SCALING WITH VM MIGRATION

In this section, we first discuss the motivation for scaling the virtual cluster with VM migration and then illustrate the detailed algorithm design.

3.1 | Benefits of VM migration

The above proposed Algorithm 1 aims to properly allocate additional VMs to the virtual cluster when scaling across geo-distributed datacenters. However, it may still turn out to be an impossible task to find a workable solution in the end, even if we actually had enough resources to accommodate such virtual cluster with a scaled size.

Algorithm 1 Scaling without VM Migration

Input:

Bandwidth requirement of the virtual cluster: B;
The original cluster size: N;
The original VM placement: $N_i, \forall i \in \mathcal{V}$;
The number of empty slots: $S_i, \forall i \in \mathcal{V}$;
Residual bandwidth of the link: $R_{i,j}, \forall l_{i,j} \in \mathcal{E}$;
Price per unit bandwidth of the link: $p_{i,j}, \forall l_{i,j} \in \mathcal{E}$;
The scaled cluster size: N';

Output:

VM placement for the scaled cluster: $N'_i, \forall i \in \mathcal{V}$;

1: Define the increase in number of VMs $\hbar = N' - N$;
2: Initialize $h = 0$ and $N'_i = N_i$ $(\forall i \in \mathcal{V})$;
3: **while** $h < \hbar$ **do**
4: Define $\mathcal{V}' \leftarrow \varnothing$ and $\tilde{N}_i = N'_i$ $(\forall i \in \mathcal{V})$;
5: **for** each datacenter $i \in \mathcal{V}$ **do**
6: **if** $S_i \geq 1$ **then**
7: Define $\tilde{N}_i = \tilde{N}_i + 1$;
8: For all $j \in \mathcal{V}$ and $j \neq i$, compute $\Delta_{i,j}^{bw} = B(\min(\tilde{N}_i, \tilde{N}_j) - \min(N_i, N_j))$.
9: **if** $\Delta_{i,j}^{bw} \leq R_{i,j}, \forall j \neq i$ **then**
10: Compute the bandwidth cost C based on Equation 2 and append datacenter i to the set $\mathcal{V}'$;
11: **end if**
12: **end if**
13: **end for**
14: Search a datacenter i in the set $\mathcal{V}'$ with the minimum bandwidth cost, and place a new VM in it, i.e., $N'_i = N'_i + 1$;
15: Update the number of empty slots of the datacenter being searched i, i.e., $S_i = S_i - 1$;
16: Update the residual bandwidth of all the links associated with i, e.g., $R_{i,j} = R_{i,j} - \Delta_{i,j}^{bw}$ $(\forall j \neq i)$;
17: h++;
18: **end while**

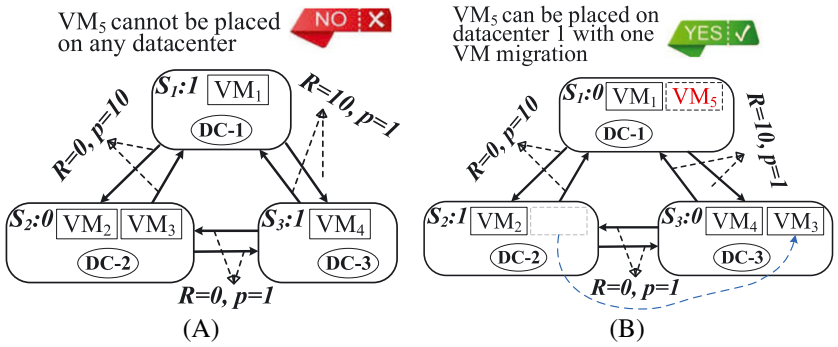


FIGURE 2 A motivating example: A, no feasible allocation exists for scaling the virtual cluster; B, the virtual cluster can be scaled if one VM (eg, VM 3) is migrated from datacenter 2 to datacenter 3

To have a comprehensive understanding, we discuss a motivating example in Figure 2, where there are 3 datacenters, and a virtual cluster similarly needs to be scaled from $< 4, 1 >$ to $< 5, 1 >$. The number of empty slots of each datacenter, the residual bandwidth, and the price per unit bandwidth of each link, as well as the current VM placement for the virtual cluster are all shown in Figure 2A. It can be easily verified that in Figure 2A, the virtual cluster is unable to be scaled because a new VM cannot be added to any datacenter. For example, if placing VM 5 in datacenter 1, then the link between datacenters 1 and 2 will be overloaded. Similarly, placing VM 5 in datacenter 3 will also lead to link congestions between datacenters 2 and 3. In addition, datacenter 2 cannot accommodate VM 5 as well due to the lack of empty slots. That is, no feasible solution can be constructed. However, in Figure 2B, it is obvious that the size of this virtual cluster can be successfully scaled to 5. This implies that the original VM placement may hinder the scalability of a virtual cluster. To tackle this issue, we leverage VM migration to optimize the placement of the original VMs to accommodate new ones. As illustrated in Figure 2B, after a VM (VM 3) is migrated from datacenter 2 to datacenter 3, we can place a new VM (VM 5) in datacenter 1 without overloading any inter-datacenter link. Therefore, we are motivated to consider the virtual cluster scaling with VM migration enabled. Given the VM migration brings service downtimes, and the performance isolation among different virtual clusters is usually expected in the cloud, nevertheless, we only consider the VM migration within the virtual cluster to be scaled. Furthermore, VM migration generates significant inter-datacenter traffic, which directly incurs high cost because of the expensive inter-datacenter bandwidth. Hence, we need to incorporate such traffic cost when designing the cluster scaling algorithm with VM migration enabled.

The key for designing the algorithm is to hypothetically remove the original virtual cluster $< N, B >$, release the bandwidth reserved on each link and the slots occupied in each datacenter, then find a valid allocation for the scaled cluster $< N', B >$. To ease the presentation, we formulate an optimization problem to guide the design of such cluster scaling algorithm with VM migration enabled. Specifically, there are two types of cost incurred by the scaled cluster: the cost of inter-datacenter bandwidth reservation C^{bw} and the cost of VM migration C^{mgr}, which are defined as follows:

$$C^{bw} = \sum_{l_{ij}} p_{ij} \min(N'_i, N'_j) * B, \tag{7}$$

$$C^{mgr} = \sum_{i} \max(N_i - N'_i, 0) * W, \tag{8}$$

where W is the cost of migrating one VM. C^{bw} is calculated as the summation of the cost of each link l_{ij}, where the term $\min(N'_i, N'_j) * B$ is the amount of bandwidth that must be reserved on each link l_{ij}, to guarantee the bandwidth requirement B for each VM-pair. In Equation 8, the term $\max(N_i - N'_i, 0)$ is the number of VMs to be moved out of datacenter i, and accordingly, C^{mgr} equals to the summation of the migration cost across all datacenters. Note that we can enforce different settings for W on different datacenters. However, it is a complicated, time-consuming, and often thankless task. It is undetermined which setting would be better and more practical, with the significant variability in both the size of VM instances, and the price per unit bandwidth across different datacenters. We believe that such unified setting can approximately reflect the cost of migrating one VM and is also appropriate to be adopted for the purpose of guiding the algorithm design with VM migration enabled.

Based on the above two types of cost, we now can formulate a new optimization problem **P2**:

$$\min_{N'_i} \quad C^{bw} + C^{mgr} \tag{9}$$

$$\text{S.t.} \quad \min(N'_i, N'_j) * B \leq \quad R_{ij} + \min(N_i, N_j) * B, \forall l_{ij}, \tag{10}$$

$$N'_i \leq S_i + N_i, \forall i, \tag{11}$$

$$\sum_{i} N'_i = N'. \tag{12}$$

Equation 9 is the objective function that incorporates both the bandwidth cost and the migration cost. Note that by adding a weight factor in front of each type of cost, any desired trade-off between these two types of cost can be achieved. For the sake of simplicity, we assume that the weight factor is 1 here. Equation 10 enforces the reserved bandwidth must be no greater than the total available bandwidth of the link, which is the sum of the corresponding residual bandwidth, and the bandwidth released by hypothetically removing of the original cluster. Similarly, Equation 11 implies that the number of VMs in each datacenter should also be limited by its total available slots after the same procedure. Finally, it is obvious that the sum of VMs being placed in each datacenter should be equal to the scaled cluster size N', as indicated by Equation 12.

3.2 | Algorithm design

Algorithm 1 can find a feasible solution if it does exist; otherwise, it fails. In this case, we terminate it and turn to a new scaling method with VM migration enabled, summarized as Algorithm 2.

Algorithm 2 starts by hypothetically removing of the original cluster, updating the available bandwidth $R_{i,j}$ and the empty slots S_i, as described in Steps 1 to 2. It can then leverage a heuristic idea to place each VM of the scaled cluster in an appropriate datacenter. Specifically, when placing each VM, it eliminates the datacenter that has insufficient empty slots or inadequate bandwidth (Steps 6-13) and then searches the remaining ones to find a datacenter at the lowest cost (including both the bandwidth cost and the migration cost) if placing one more VM in it (Step 14). Finally, the algorithm updates the number of empty slots of each datacenter (Step 15) as well as the residual bandwidth on each inter-datacenter link (Step 16) then continues to allocate another VM until all VMs of the scaled cluster have been placed.

Algorithm 2 Scaling with VM Migration

Input:

Bandwidth requirement of the virtual cluster: B;
The original VM placement: $N_i, \forall i \in \mathcal{V}$;
The number of empty slots: $S_i, \forall i \in \mathcal{V}$;
Residual bandwidth of the link: $R_{i,j}, \forall l_{i,j} \in \mathcal{E}$;
Price per unit bandwidth of the link: $p_{i,j}, \forall l_{i,j} \in \mathcal{E}$;
The scaled cluster size: N';
Cost of migrating one VM: W;

Output:

VM placement of the scaled cluster: $N'_i, \forall i \in \mathcal{V}$;

1: $R_{i,j} \leftarrow R_{i,j} + \min(N_i, N_j) * B, \forall l_{i,j}$;
2: $S_i \leftarrow S_i + N_i, \forall i$;
3: Initialize $h = 0$ and $N'_i = 0, \forall i \in \mathcal{V}$;
4: **while** $h < N'$ **do**
5: Define $\mathcal{V}' \leftarrow \varnothing$ and $\tilde{N}_i = N'_i$ $(\forall i \in \mathcal{V})$;
6: **for** each datacenter $i \in \mathcal{V}$ **do**
7: **if** $S_i \geq 1$ **then**
8: $\tilde{N}_i \leftarrow \tilde{N}_i + 1$;
9: **if** $\min(\tilde{N}_i, \tilde{N}_j) * B \leq R_{i,j}, \forall j \neq i$ **then**
10: Compute the total cost $C = C^{bw} + C^{mgr}$ based on Equation 7, 8, and append datacenter i to $\mathcal{V}'$;
11: **end if**
12: **end if**
13: **end for**
14: Search a datacenter i in $\mathcal{V}'$ with the minumum C, then place a new VM in it, i.e., $N'_i = N'_i + 1$;
15: Update the number of empty slots of the datacenter i being searched, i.e., $S_i = S_i - 1$;
16: Update the residual bandwidth of all links associated with i, e.g., $R_{i,j} = R_{i,j} - (\min(N'_i, N'_j) - \min(N'_i - 1, N'_j)) * B, \forall j \neq i$;
17: h++;
18: **end while**

Discussions: We discuss two interesting issues concerning the above algorithm. *First*, it may also not work when there are insufficient resources, eg, empty slots or available bandwidth to accommodate all VMs of the scaled cluster. In this case, our algorithm will strive to place as many VMs as possible while satisfying all the guaranteed bandwidth requirements of the placed VMs. *Second*, our algorithm generates N'_i as output. It is necessary to practically place new VMs as well as migrate the original VMs to archive the desired result. This brings us to a new question: *How to migrate VMs with the VM placement for both the original cluster and the scaled cluster?* To further reduce the migration cost, we take advantage of the variability in the price per unit bandwidth to design an efficient migration strategy.

4 | PERFORMANCE EVALUATION

In this section, we investigate the performance of our proposed scaling algorithms from 3 different perspectives: the bandwidth cost of inter-datacenter links, the acceptance rate of scaling requests, and the cost of VM migrations.

4.1 | Experimental settings

Experiment Setup: In our experiment, we decide to use MATLAB environment as a simulator and create an inter-datacenter network with 30 datacenters, which is commonly seen in typical cloud networks, eg, Microsoft.[9] The number of available slots of each datacenter is set to be a uniform random integer within the range [10, 100]. In our experiments, we vary the bandwidth of inter-datacenter links between 1*Gbps* and 10*Gbps*, hoping to mimic the heterogeneous bandwidth environment across geo-distributed datacenters. In fact, such environment can be achieved by utilizing Linux Traffic Control.[21] We develop a tiered structure of the price per unit bandwidth, where an inter-datacenter link with larger bandwidth capacity has a relatively lower unit cost. To demonstrate an exact pricing relation, we choose Amazon EC2 Data Transfer Pricing.[22] For instance, if the data transfer price is \$0.0012/Mb, then the price per unit bandwidth is assumed to be \$0.0012/Mbps accordingly. Table 2 summarizes such tiered bandwidth pricing relation in detail.

Evaluation Methodology: To evaluate the performance of our proposed scaling algorithms, we mainly focus on the following 3 performance metrics: *the bandwidth cost of inter-datacenter links*, *the acceptance rate of scaling requests*, and *the cost of VM migrations*. To ease the presentation, we denote the scaling algorithm in Section 2 as "Scaling-wo-M," and the algorithm in Section 3 as "Scaling-w-M." Similarly, we use "Scaling-wo-BwC" to represent the scaling algorithm in Yu and Cai[18] as a baseline, which exploits VM migration to scale up virtual clusters within a single datacenter, while simply ignoring the bandwidth cost when performing the scaling.

We consider that there are 10 jobs. Each job specifies a virtual cluster $< N, B >$. Initially, we deploy only one VM in each datacenter for each job. We randomly generate 200 scaling requests, each request attempts to scale up the virtual cluster of one job with the same increase ratio θ. We handle these scaling requests in a first-in-first-out (FIFO) manner. The cost of migrating one VM across different datacenters is always set to 1, ie, $W = 1$.

4.2 | Experiment results

4.2.1 | Impact of parameters

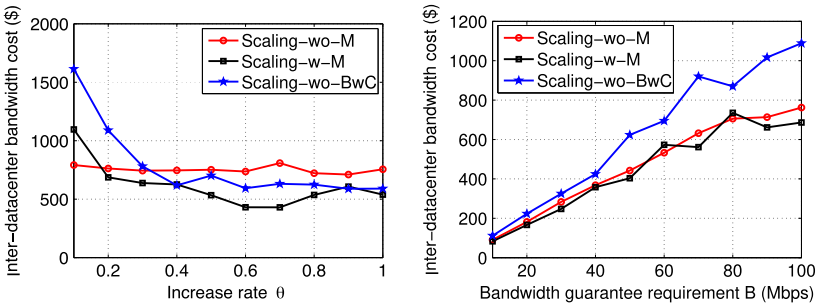
We first investigate the impact of the parameters, ie, the bandwidth requirement per virtual cluster B and the increase rate of a virtual cluster θ, on the performance of our proposed scaling algorithms. To this end, we first vary the increase rate θ from 0.1 to 1, meanwhile keeping the guaranteed bandwidth B as 100 Mbps. Then we vary B from 10 to 100 Mbps, meanwhile keeping θ as 0.2.

Figure 3A first shows the bandwidth cost variation under different increase rate θ. It is clear that the bandwidth cost of Scaling-w-M is lower than that of Scaling-wo-M, with most of θ's. The reason is that Scaling-w-M hypothetically removes the original cluster and reallocates all VMs in the scaled cluster. This significantly increases the possibility of finding an optimized solution with reduced bandwidth cost. We can further observe that Scaling-w-M always achieves a lower bandwidth cost, comparing with Scaling-wo-BwC. This is because though Scaling-wo-BwC considers the migration cost, it simply ignores the bandwidth cost when scaling up a virtual cluster. We further evaluate the impact of guaranteed bandwidth requirement B on the bandwidth cost, as illustrated in Figure 3B, under fixed increase rate $\theta = 0.2$. One can easily check that the bandwidth cost of all these three algorithms (Scaling-wo-M, Scaling-w-M, and Scaling-wo-BwC) increases along with the growth of guaranteed bandwidth requirement B. Moreover, under all settings of B, Scaling-wo-BwC is the worst in minimizing the bandwidth cost, and for most of B's, Scaling-w-M achieves a little bit lower bandwidth cost than Scaling-wo-M.

It should be noted that the Scaling-w-M algorithm may be effective in reducing the bandwidth cost of inter-datacenter links, while as a consequence, it also introduces a significant amount of migration costs. To quantitatively formulate this issue, Figure 4 presents the migration cost of Scaling-w-M and Scaling-wo-BwC. Specifically, Figure 4A and 4B plots the migration cost with various parameters, θ and B, respectively. One can easily check that the migration cost of these two algorithms decreases with the increasing of both θ and B. Furthermore, one may wonder at this point that Scaling-wo-BwC achieves a lower migration cost than the proposed Scaling-w-M. This is because Scaling-wo-BwC scales up a virtual cluster with only a single objective of minimizing the migration cost. That is why Scaling-wo-BwC incurs almost the top bandwidth costs as illustrated in

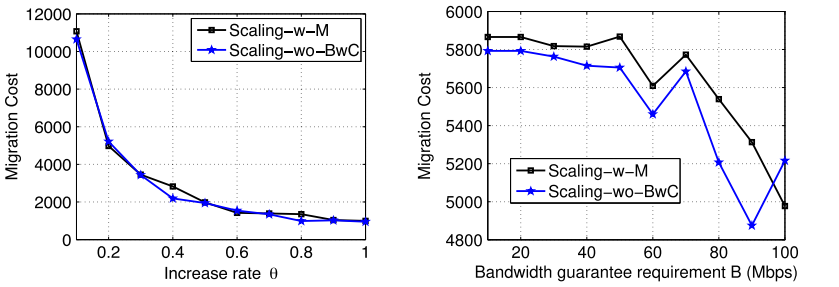
TABLE 2 Tiered bandwidth pricing

Bandwidth capacity, Mbps	Price, \$/Mbps
< 2000	0.0012
$2000 - 4000$	0.0009
$4000 - 6000$	0.0007
$6000 - 8000$	0.0005
> 8000	0.0003



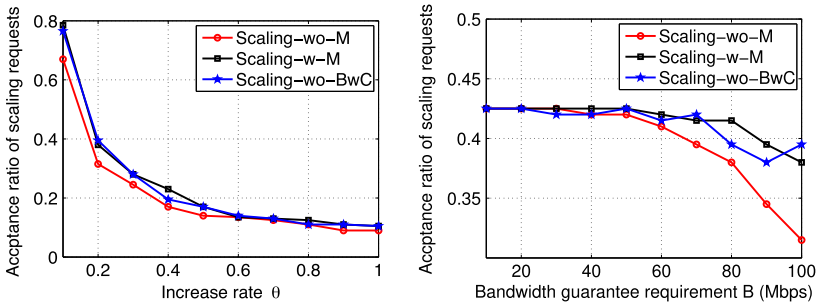
(A) Varying rate θ, with fixed $B = 100$Mbps (B) Varying bandwidth B, with fixed $\theta = 0.2$

FIGURE 3 The performance on bandwidth cost achieved by Scaling-wo-M, Scaling-w-M, and Scaling-wo-BwC algorithms: A, the bandwidth cost vs the increase rate θ, with fixed bandwidth $B = 100$ Mbps; B, the bandwidth cost vs the guaranteed bandwidth B, with fixed rate $\theta = 0.2$



(A) Varying rate θ, with fixed bandwidth $B = 100$Mbps (B) Varying bandwidth B, with fixed rate $\theta = 0.2$

FIGURE 4 The performance on migration cost of Scaling-w-M and Scaling-wo-BwC algorithms: A, the migration cost vs the increase rate θ, with fixed bandwidth $B = 100$ Mbps; B, the migration cost vs the guaranteed bandwidth B, with fixed rate $\theta = 0.2$



(A) Varying rate θ, with fixed $B = 100$Mbps (B) Varying bandwidth B, with fixed $\theta = 0.2$

FIGURE 5 The performance on acceptance rate of scaling-wo-M and scaling-w-M algorithms: A, the acceptance rate of scaling requests vs the increase rate θ, with fixed bandwidth $B = 100$ Mbps; B, the acceptance rate of scaling requests vs the guaranteed bandwidth B, with fixed rate $\theta = 0.2$

Figure 3. It should also be noted that one can easily make the migration cost of Scaling-w-M arbitrarily close to that of Scaling-wo-BwC, by choosing appropriate parameters θ and B, eg, $B = 100$ Mbps and $\theta = 0.2$.

One of the most important metrics when scaling the cluster across multiple datacenters is the guaranteed bandwidth requirement. For each scaling request, it is acceptable only if the guaranteed bandwidth requirement of each VM-pair can be satisfied. The acceptance rate is then calculated as the number of accepted requests dividing by the number of all requests has been raised. Figure 5A first demonstrates the acceptance rate of scaling requests with various θ. It is clear that the acceptance rate decreases with the increasing of θ. We further observe that the acceptance rate of Scaling-w-M is higher than that of Scaling-wo-M, irrespective of any variety in θ, and the acceptance rate of Scaling-wo-BwC is in the middle. Towards a more comprehensive understanding, we further plot the acceptance rate under different settings of guaranteed bandwidth requirement B, as illustrated in Figure 5B. We also notice that in all these three algorithms, the acceptance rate remains almost unchanged when $B \leq 30$ Mbps. However, Scaling-w-M can accommodate more scaling requests than the remaining two algorithms, when $B > 30$ Mbps.

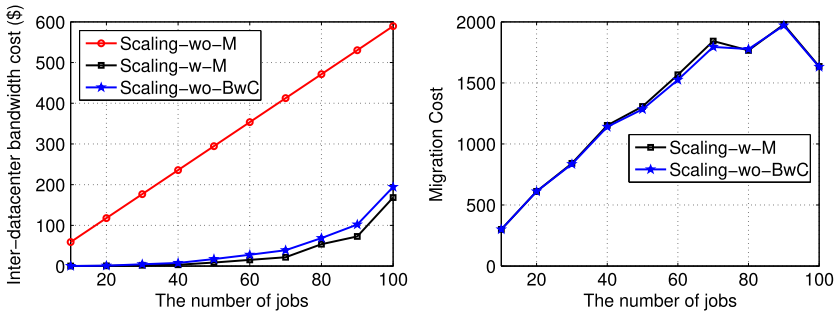


FIGURE 6 The impact of the number of jobs: A, the bandwidth cost vs the number of jobs; B, the migration cost vs the number of jobs

TABLE 3 The bandwidth cost, migration cost, and acceptance rate of scaling requests when handling multiple requests under different scaling algorithms

	Bandwidth cost ($)	Migration cost ($)	Acceptance rate of scaling requests
Original scaling-wo-M	751.2	0	0.14
Combining SCF and scaling-wo-M	742.4	0	0.175
Original scaling-w-M	534.1	1982	0.17
Combining SCF and scaling-w-M	401.5	2322	0.2

4.2.2 | Impact of the number of jobs

In the above experiments, only 10 jobs are running. We now evaluate the impact of the number of jobs on the performance of proposed scaling algorithms. To this end, we vary the number of jobs from 10 to 100, then run the experiment 10 rounds. We randomly generate the scaling requests, with the number of requests equalling jobs each round. Across all rounds of experiments, we set $B = 10$ Mbps and $\theta = 0.1$. Figure 6A first plots the bandwidth cost incurred by Scaling-wo-M, Scaling-w-M, and Scaling-wo-BwC algorithms, under different number of jobs. Obviously, the bandwidth cost incurred by all these algorithms increases along with the growth of the number of jobs, and Scaling-wo-M algorithm performs the worst saving the bandwidth cost. We further observe that Scaling-w-M can achieve a lower bandwidth cost than Scaling-wo-Bwc, irrespective of the number of jobs changing. Figure 6B further plots the migration cost introduced by Scaling-w-M and Scaling-wo-BwC algorithms, through varying the number of jobs. We can see that even though Scaling-wo-BwC aims only at minimizing the migration cost, Scaling-w-M achieves almost the same migration cost with Scaling-wo-BwC, across all settings of the number of jobs. This implies that our Scaling-w-M can leverage VM migration to scale up a virtual cluster in a more cost-effective way. Note that all requests in each round of experiment have been accepted; we therefore do not show the acceptance rate of these scaling requests further.

4.2.3 | Handling concurrent scaling requests

So far, we have only focused on the scenario where the scaling requests are handled one by one with FIFO scheduling policy. However, in most real-world cases, there are typically more than one job submitting scaling requests concurrently. Simply scheduling these requests with FIFO is insufficient to minimize the bandwidth cost as well as the migration cost. Hence, we need to decide which virtual cluster to be scaled each time. To this end, we apply a shortest-cost-first (SCF) scheduling policy. In other words, each time when we are ready to handle a request, we sort all requests that have not been served in an ascending order based on the cost each will incur. Then we select a request with the lowest cost and scale up its corresponding virtual cluster with the proposed Scaling-wo-M and Scaling-w-M algorithms. With regard to the experiment, we still generate 10 jobs and 200 requests, with fixed $B = 100$ Mbps and $\theta = 0.5$. Then these 200 scaling requests are handled by four algorithms: (1) the original Scaling-wo-M with FIFO, (2) the algorithm by combining SCF with Scaling-wo-M, (3) the original Scaling-w-M with FIFO, and (4) the algorithm by combining SCF scheduling with Scaling-w-M. Table 3 presents the bandwidth cost, migration cost, and acceptance rate under all four algorithms. We can easily check that when combining the SCF scheduling with scaling-wo-M, the bandwidth cost can be reduced to 742.4 and the acceptance rate of requests can be increased to 0.175. On the other hand, combining the SCF scheduling with scaling-w-M, the bandwidth cost can be reduced further; meanwhile, the migration cost would also be increased at the same time, due to a larger acceptance rate. The above results demonstrate that when there are more than one concurrent scaling request, the methods of request scheduling and virtual cluster scaling should be jointly considered to optimize the overall costs.

5 | DISCUSSION

Taking energy into consideration: Energy expenses account for a large potion of the overall datacenter operation costs.[23,24] To incorporate it into our proposed algorithms, a simple approach is to leverage the varying energy prices across different datacenters when scaling up a virtual cluster.

Furthermore, it would be even better to leverage the time-varying energy prices[24] to decide when to scale dynamically. We leave this as one direction of our future work.

Avoiding frequent scaling-up/scaling-down: In this paper, we only focus on scaling up a virtual cluster. However, since the workload of the upper-level applications changes over time, the virtual cluster also needs to be scaled down when necessary. To avoid frequent scaling-up/scaling-down, we can introduce kind of adaptive damping mechanism to limit the number of scaling requests permitted at any given duration, as another direction of our future work.

Reducing the cost of migration over wide-area networks: Migrating VMs over the wide-area network (WAN) could incur high cost, because WAN bandwidth is always limited. One possible approach would be to combine a block-level solution with pre-copying and write throttling.[25] We can also use CloudNet, which provides a set of optimizations that minimize the cost of storage and VM memory transfer over low-bandwidth and high-latency WAN links during the migration.[26] We leave this as an open issue in the literature.

6 | RELATED WORK

The proposed scaling algorithms mainly focus on allocating network resources of datacenters to virtual clusters. There is a large spectrum of related work. Here, we only review the most closely related ones in the fields of network resource sharing either within a single datacenter, or across geo-distributed datacenters, respectively.

6.1 | Network resource sharing within a single datacenter

Towards the resource sharing among virtual clusters within a single datacenter, the proposed methods so far can be roughly classified into two categories: reserving network resources to virtual clusters with fixed requirements and reserving network resources to virtual clusters with varying requirements. Regarding the former category, SecondNet[2] first presents VDC (virtual data center) as an abstraction for describing the bandwidth requirement between each VM-pair and allocates bandwidth via the hypervisor of servers to provide bandwidth guarantees at VM-to-VM level. Lee et al propose a new network abstraction TAG (tenant application graph) and execute a workload placement algorithm to guarantee the bandwidth specified by TAGs.[5] Oktopus[3] designs a virtual cluster model and leverages an efficient VM placement algorithm, as well as a static rate limit to enforce the bandwidth guarantee on the virtual cluster. However, Oktopus ignores the highly variable demand on the bandwidth by the VMs. With such insight, Xie et al[4] suggest a TIVC (temporally interleaved virtual cluster) model, which makes time-varying bandwidth reservation based on different bandwidth specifications during different time intervals. In addition to the network abstraction model, some other solutions focus on providing the minimum bandwidth guarantees. For example, EyeQ[27] presents such method by reserving a minimum bandwidth for each endpoint on resource provisioning. Gatekeeper[28] leverages a distributed mechanism to reserve the bandwidth for VMs of a tenant. Although all the above methods can provide bandwidth guarantee to the virtual cluster, they do not consider the changes of the resource requirements of a virtual cluster. This urges the elastic resource sharing, which dynamically scales the resources of a VM or cluster so as to adapt to workload changes of cloud applications. For example, Pedala et al design a feedback control system to dynamically make reservations of CPU and disk resources for VMs, with overlooking the network resources.[29] Instead of the feedback mechanism, Gong et al perform the resource scaling based on a lightweight online prediction scheme.[30] For the resource scaling at cluster-level, Nguyen et al propose AGILE, which dynamically adjusts the number of VMs being allocated to a cloud application based on its SLO (Service Level Objectives).[16] Taking it one step further, Herodotou et al target on automatically determining the cluster size as well as the instance type of VMs, to archive the desired performance for a given workload.[17] Han et al jointly consider the resource scaling at both VM-level and cluster-level.[15] Those scaling methods do not consider the guaranteed bandwidth requirement in general. To tackle this issue, Yu et al address the bandwidth guarantees problem of virtual cluster scaling,[18] while their method is only applicable to the intra-datacenter scenario. Niu et al focus on dynamically booking a minimum bandwidth from multiple cloud providers for video-on-demand applications, by designing a predictive resource auto-scaling system.[1] However, they actually reserve the intra-datacenter bandwidth from different cloud providers, far from the inter-datacenter scenario concerned.

In addition to resource reserving, there are also a large number of existing methods that allocate datacenter network resources to VMs or virtual clusters after the placements have been fixed, [14,31-33] or schedule the workflow of applications running on top of these virtual clusters.[34]

6.2 | Network resource sharing across geo-distributed datacenters

Regarding network resource sharing across geo-distributed datacenters, Li et al[35] propose an ADMM-based algorithm to provide bandwidth guarantees to inter-datacenter traffic at the minimum bandwidth cost. The ADMM-based algorithm can work in conjunction with our proposals. This is because that ADMM-based algorithm targets on optimizing the transmission of inter-datacenter traffic in a bandwidth-guaranteed manner by assuming fixed VM placement of an application, while our proposed algorithms focus on scaling up the resources of an application to adapt to its workload changes instead.

There are a large body of recent researches aim at reducing the cost of inter-datacenter traffic. For example, Feng et al propose to optimally routing the inter-datacenter video traffic to minimize the cloud provider's operation costs. Liu et al[36] propose efficient data replication scheme

to reduce the amount of inter-datacenter traffic, therefore save the traffic cost accordingly. Laoutaris et al[37] present efficient store-and-forward method for inter-datacenter bulk data transfers.[38]

Regarding energy or carbon reduction, Khosravi et al[23] take advantages of the diverse carbon footprint rates and PuEs across different datacenters to design a novel VM placement algorithm. Their algorithm can efficiently reduce the CO_2 emission and power consumption. Lin et al[39] propose a two-time-scale Lyapunov optimization–based approach to dynamically make decisions on the number of active servers based on the incoming workload, so as to save significant amount of energy.

Regarding the cost of VM migration, Voorsluys et al[40] investigate the impact of live VM migration on the performance of applications running on top of the VM. Their results demonstrate that the migration overhead is acceptable in most cases.

7 | CONCLUSION

This paper takes a first step towards addressing the problem of scaling up a virtual cluster across geo-distributed datacenters, with the aim of reducing the bandwidth cost of inter-datacenter links as well as fulfilling the guaranteed bandwidth requirement for each VM-pair of the cluster. We formulate the problem of virtual cluster scaling as an optimization problem. Specifically, we present two scaling algorithms: one is to scale up a virtual cluster with the original VM placement unchanged, and the other is to scale the cluster with VM migration enabled. We have conducted extensive simulations to demonstrate the performance of the proposed algorithms. The results verify that our algorithms are effective in both the bandwidth cost minimization and the guaranteed bandwidth requirement satisfaction. The results further demonstrate that the scalability of the virtual cluster can be significantly improved with VM migration enabled.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China No. 2016YFB1000205; the State Key Program of National Natural Science of China No. 61432002; NSFC Grant Nos 61772112, 61272417, 61300189, 61370199 and 61672379.

ORCID

Xinping Xu http://orcid.org/0000-0003-4545-4236
Heng Qi http://orcid.org/0000-0002-8770-3934

REFERENCES

1. Niu D, Xu H, Li B, Zhao S. Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In: Proceedings of IEEE INFOCOM; 2012; Orlando, USA. 460-468.
2. Guo C, Lu G, Wang HJ, et al. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In: Proceedings of acm CoNext; 2010; Philadelphia, USA. 15-15.
3. Ballani H, Costa P, Karagiannis T, Rowstron A. Towards predictable datacenter networks. In: Proceedings of ACM SIGCOMM; 2011; Toronto, ON, Canada. 242-253.
4. Xie D, Ding N, Hu YC, Kompella R. The only constant is change: incorporating time-varying network reservations in data centers. In: Proceedings of ACM SIGCOMM; 2012; Helsinki, Finland. 199-210.
5. Lee J, Turner Y, Lee M, et al. Application-driven bandwidth guarantees in datacenters. In: Proceedings of ACM SIGCOM; 2014; Chicago, USA. 467-478.
6. Jain S, Kumar A, Mandal S, et al. B4: Experience with a globally-deployed software defined WAN. In: Proceedings of ACM SIGCOMM; 2013; Hong Kong, China. 3-14.
7. Li W, Guo D, Li K, Qi H, Zhang J. iDaaS: inter-datacenter network as a service. *IEEE Trans Parallel Distrib Syst*. 2015. https://doi.org/10.1109/TPDS.2015.2505731.
8. Li W, Qi H, Li K, Stojmenovic I, Lan J. Joint optimization of bandwidth for provider and delay for user in software defined data centers. In: *IEEE Trans Cloud Comput*. 2017;5(2):331-343.
9. Hong CY, Kandula S, Mahajan R, et al. Achieving high utilization with software-driven wan. In: Proceedings of ACM SIGCOMM; 2013; Hong Kong, China. 15-26.
10. Chen F, Guo K, Lin J, La Porta T. Intra-cloud lightning: building CDNs in the cloud. In: Proceedings of IEEE INFOCOM; 2012; Orlando, America. 433-441.
11. Adhikari VK, Guo Y, Hao F, et al. Unreeling netflix: understanding and improving multi-cdn movie delivery. In: Proceedings of IEEE INFOCOM; 2012; Orlando, America. 1620-1628.
12. Chen Y, Jain S, Adhikari VK, Zhang ZL, Xu K. A first look at inter-data center traffic characteristics via yahoo! datasets. In: Proceedings of IEEE INFOCOM; 2011; Shanghai, China. 1620-1628.
13. Pu Q, Ananthanarayanan G, Bodik P, et al. Low latency geo-distributed data analytics. In: Proceedings of ACM SIGCOMM; 2015; London, United Kingdom. 421-434.
14. Guo J, Liu F, Zeng D, Lui J, Jin H. A cooperative game based allocation for sharing data center networks. In: Proceedings of IEEE INFOCOM; 2013; Turin, Italy. 2139-2147.
15. Han R, Guo L, Ghanem MM, Guo Y. Lightweight resource scaling for cloud applications. In: Proceedings of IEEE CCGRID; 2012; Ottawa, ON, Canada. 644-651.

16. Nguyen H, Shen Z, Gu X, Subbiah S, Wilkes J. Agile: elastic distributed resource scaling for infrastructure-as-a-service. In: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13); 2013; San Jose, USA. 69-82.
17. Herodotou H, Dong F, Babu S. No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics. In: Proceedings of the 2nd ACM Symposium on Cloud Computing; 2011; Cascais, Portugal. 18-18.
18. Yu L, Cai Z. Dynamic scaling of virtual clusters with bandwidth guarantee in cloud data centers. In: Proceedings of IEEE INFOCOM; 2016; San Francisco, USA. 1-9.
19. The pricing of data transfer on amazon ec2 platform. https://aws.amazon.com/cn/ec2/pricing/on-demand/. Accessed on February 12, 2017.
20. Dantzig GB. *Linear Programming and Extensions*. Princeton, America: Princeton University Press; 1998.
21. Linux traffic control. http://lartc.org/manpages/tc.txt. Accessed on February 12, 2017.
22. Xu H, Li B. Joint request mapping and response routing for geo-distributed cloud services. In: Proceedings of IEEE INFOCOM; 2013; Turin, Italy. 854-862.
23. Khosravi A, Garg SK, Buyya R. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In: Proceesings of Springer European Conference on Parallel Processing; 2013; Heidelberg, Germany. 317-328.
24. Qureshi A, Weber R, Balakrishnan H, Guttag J, Maggs B. Cutting the electric bill for internet-scale systems. In: Proceedings of ACM SIGCOMM; 2009; Barcelona, Spain. 123-134.
25. Bradford R, Kotsovinos E, Feldmann A, Schiöberg H. Live wide-area migration of virtual machines including local persistent state. In: Proceedings of the 3rd ACM International Conference on Virtual Execution Environments; 2007; San Diego, USA. 169-179.
26. Wood T, Ramakrishnan KK, Shenoy P, Van der Merwe J. Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines. In: Proceedings of ACM Sigplan Notices; 2011; Newport Beach, USA. 121-132.
27. Jeyakumar V, Alizadeh M, Mazieres D, Prabhakar B, Kim C, Greenberg A. Eyeq: practical network performance isolation at the edge. In: Proceedings of USENIX NSDI; 2013; Lombard, IL, USA. 297-311.
28. Rodrigues H, Santos JR, Turner Y, Soares P, Guedes D. Gatekeeper: supporting bandwidth guarantees for multi-tenant datacenter networks. In: USENIX WIOV; 2011; Portland, OR, USA. 6-6.
29. Padala P, Hou KY, Shin KG, et al. Automated control of multiple virtualized resources. In: Proceedings of ACM European Conference on Computer Systems; 2009; Nuremberg, Germany. 13-26.
30. Gong Z, Gu X, Wilkes J. Press: predictive elastic resource scaling for cloud systems. In: Proceedings of IEEE International Conference on Network and Service Management; 2010; Niagara Falls, ON, Canada. 9-16.
31. Shieh A, Kandula S, Greenberg A, Kim C, Saha B. Sharing the data center network. In: Proceedings of USENIX NSDI; 2011; Boston, America. 23-23.
32. Popa L, Kumar G, Chowdhury M, Krishnamurthy A, Ratnasamy S, Stoica I. Faircloud: sharing the network in cloud computing. In: Proceedings of ACM SIGCOMM; 2012; Helsinki, Finland. 187-198.
33. Chen L, Feng Y, Li B, Li B. Towards performance-centric fairness in datacenter networks. In: Proceedings of IEEE INFOCOM; 2014; Toronto, Canada. 1599-1607.
34. Stavrinides GL, Karatza HD. A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds. In: Proceedings of the 3rd IEEE International Conference on Future Internet of Things and Cloud (FICLOUD); 2015; Rome, Italy. 231-239.
35. Li W, Li K, Guo D, Min G, Qi H, Zhang J. Cost-minimizing bandwidth guarantee for inter-datacenter traffic. *IEEE Trans Cloud Comput*. 2016. https://doi.org/10.1109/TCC.2016.2629506.
36. Liu G, Shen H, Chandler H. Selective data replication for online social networks with distributed datacenters. In: Proceedings of IEEE ICNP; 2013; Goettingen. 2377-2393.
37. Laoutaris N, Sirivianos M, Yang X, Rodriguez P. Inter-datacenter bulk transfers with netstitcher. In: Proceedings of ACM SIGCOMM; 2011; Toronto, Canada. 74-85.
38. Feng Y, Li B, Li B. Jetway: minimizing costs on inter-datacenter video traffic. In: Proceedings of ACM Multimedia; 2012; Nara, Japan. 259-268.
39. Lin M, Wierman A, Andrew LLH, Thereska E. Dynamic right-sizing for power-proportional data centers. *IEEE/ACM Trans Netw (TON)*. 2013;21(5):1378-1391.
40. Voorsluys W, Broberg J, Venugopal S, Buyya R. Cost of virtual machine live migration in clouds: a performance evaluation. *CloudCom*. 2009;9:254-265.

How to cite this article: Xu X, Li W, Qi H, Li K. On efficient virtual cluster scaling across geo-distributed data centers. *Concurrency Computat Pract Exper*. 2018;30:e4383. https://doi.org/10.1002/cpe.4383