Elastic Scheduling for Scaling Virtual Clusters in Cloud Data Center Networks

SHUAIBING LU¹,² (Member, IEEE), ZHIYI FANG¹, JIE WU² (FELLOW, IEEE), AND GUANNAN QU¹.
¹College of Computer Science and Technology, Jilin University, Changchun, 130012, China (e-mail: lushuaibing11@163.com, fangzy@jlu.edu.cn, gnuq@jlu.edu.cn)
²Center for Networked Computing, Temple University, USA (e-mail: jiewu@temple.edu)
Corresponding author: Shuaibing Lu (e-mail: lushuaibing11@163.com).

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

ABSTRACT Data Center Networks (DCNs) have become more extensively applied in cloud computing in recent years. One important mission for DCNs is satisfying the fluctuation of on-demand resources for tenants. During the scaling of Virtual Clusters (VCs), existing works fail to fully consider placement techniques and the elasticity of the physical resource in the DCN at the same time. To address this, we use elasticity to measure the scaling potential of VCs in terms of both computation and communication resources. In this paper, we consider elastic scaling for existing VCs to maximize elasticity with the constraint of communication cost in the DCN. We achieve this through a resource allocation scheme, VCS, which comes with provable optimality guarantees for single VC scaling. After that, we extend our scheme for the scaling of multiple VCs, and we prove that scaling multiple VCs for the over-time elasticity maximization problem is NP-hard. Based on that, we present the MVCS algorithm for offline multiple VC scaling, which can maximize over-time elasticity during a stable time period. Furthermore, we propose two heuristic algorithms, S-OMVCS and A-OMVCS, using Bayesian parameter estimation to solve an online scaling with both synchronous and asynchronous incoming rates. Extensive simulations demonstrate that our elastic VC scaling placement schemes outperform existing state-of-the-art methods in terms of flexibility in the DCN.

INDEX TERMS Data center networks (DCNs), elastic scaling, Virtual Cluster (VC), optimization, Virtual Machine (VM) placement.

I. INTRODUCTION

Data Center Networks (DCNs) have become more extensively applied in cloud computing in recent years. Applications based on the cloud generate a significant amount of network traffic, and a considerable fraction of their runtime is due to network activity [1]. As reported in [2], [3], the resource available to tenants varies over time in EC2. One major problem for cloud computing tenants is lack of performance guarantee, which includes both resource limitations and unpredictable application demands.

To illustrate this, we show the average CPU utilization for the word count application in Hadoop (based on EC2 of Amazon [4]). We use four nodes to run this application: three slaves (orange, light green, and green lines) and one master (blue line). It is a three hours tracing result of the CPU utilization for these three slaves. We observe from Figure 1 that the word count application utilizes the CPU of the master node only during the map part of the execution (during the shuffle phase of the blue line). The three slaves (reduce phase) require very little resource. For each slave, the amount of resources required is time-varying. Since the configurations of slaves are different, resource inefficiencies may be encountered during the running period, as shown in the last hour in Fig. 1. Therefore, efficiently provisioning network resource for elastic scaling virtual clusters in the cloud is a critical issue.

We propose an elastic placement strategy to deal with resource scaling for the existing Virtual Clusters (VCs) in the DCN. A set of Virtual Machines (VMs) that connect one virtual switch is defined as a VC. Each VC not only has computing requirements but also requires communications among itself to complete the specified tasks for the applica-
We use elasticity [5] to measure the growth potential of a VC placement for both computation and communication. We denote this as Physical Machine (PM) elasticity and Physical Link (PL) elasticity. Our objective is to maximize the elasticity during the placement of the VCs’ scaling requests in the DCN under the resource and communication cost constraints. This paper is based on existing DCN architecture: a fat-tree. The capacities of PMs are slotted, and each slot can only host one VM, as shown in Fig. 2. The communication between VMs occurs through the PLs of each VC. The corresponding communication demands are determined by VM communication models. This paper uses the hose model, a communication model used to calculate bandwidth demand for VMs.

To maximize the elasticity in the DCN under the hose communication model, we face one important challenge: balancing the trade-off between communication cost and elasticity during the VC placement while guaranteeing both computation and communication resource scaling for one VC under the DCN. Take Fig. 2 as an example; VC1 has 7 existing VMs. We assume the upper bound of the subtree root is in the aggregation switch level. The scaling request for VC1 is from 5 VMs. One extreme assignment for the scaling request is to concentrated place all the scaling VMs into the PMs $M_1$ and $M_2$ under the switch $S_{11}$, as shown in Fig. 2 (a). This solution can save communication cost between existing VMs and incoming ones. However, the elasticity of VC1 decreases to 0, creating two bottleneck PMs, $M_1$ and $M_2$, with no scaling potential. Another extreme assignment is placing the VMs dispersedly, as shown in Fig. 2 (b). In this case, the elasticity of VC1 will be $\min\{\frac{2}{5}, \frac{2}{5}, \frac{3}{5}, \frac{3}{5}, \frac{4}{5}, \frac{4}{5}\} = \frac{2}{5}$. However, the communication cost under this assignment has already moved beyond the upper bound of the subtree root, which cannot guarantee QoS for the users. To balance the trade-off between communication cost and elasticity, we try to find a solution between the two extreme assignments. In this paper, we propose an optimal solution that makes a placement based on the proportion of the remaining available capacities for the scaling request under the limitation of communication cost, as shown in Fig. 2 (c). Then, we extend our solution to the online multiple VC scaling placement problem, which is solved by a heuristic method with prediction. Our algorithm can improve over-time elasticity using historical knowledge and distribution to place the current scaling VCs.

In this paper, we jointly consider the placement and elasticity adjustment problems for scaling VCs to maximize elasticity with the constraint of communication cost in the DCN. Our contributions can be summarized as follows:

- We show that there is a trade-off between elasticity and communication cost for a VC scaling request. Given one scaling request, the decreasing placement of the elasticity may lead to an increase in the communication cost. We prove the bound for the extra cost, and we discuss the existence of an optimal solution during the elasticity adjustment.
- We propose an algorithm, VCS, for the scaling request of an existing VC under the constraints of resource and communication costs, and we prove that it is an optimal solution.
- We extend the single VC scaling placement problem for multiple VCs. We also prove that it is NP-hard, and we propose the MVCS algorithm for scaling resource during a stable time period to maximize the over-time elasticity of VCs.
- We describe the online condition for the multiple VC scaling problem with both synchronous and asynchronous incoming rates, and we propose two heuristic algorithms, S-OMVCS and A-OMVCS using the Bayesian parameter estimation.
- We conduct various simulations to compare our joint optimization method with other state-of-art approaches. The results are shown from different perspectives to provide conclusions.

The remainder of this paper is organized as follows. Section II surveys related works. Section III describes the model and then formulates the problem. Section IV investigates...
II. RELATED WORK

There are tremendous works about resource allocation for VC scaling. It is a technique of crucial importance, which means that researchers must find appropriate embedding for virtual clusters in DCNs. This section provides a brief overview of the relevant methodologies proposed to address this problem. Since most research focuses on dynamically adjusting cluster size without considering bandwidth guarantees targeted by current network abstractions, several methods have been proposed. [6] proposes scaling a virtual network abstraction with a bandwidth guarantee. Efficient algorithms are proposed to find a valid allocation for the scaled cluster abstraction with optimization on the VM locality of the cluster. [7] proposes a virtual cluster abstraction called Stochastic Virtual Cluster (SVC) to realize bandwidth guarantee during the resource allocation. The framework and algorithms ensure that the bandwidth demands of tenants on a link are satisfied with a high probability while minimizing the bandwidth occupancy cost on links.

Elasticity has been considered one of the central attributes of cloud computing [8]. 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 [9]. In order to define a measure of the elasticity, [10] provides a set of benchmarks for cloud computing performance. Elastic resource scaling has attracted considerable attention in cloud computing [5], [11]. [12] proposes a lightweight approach to enable cost-effective elasticity for cloud applications; this is realized by designing an automatic system. There are also a number of works about scaling resources using a prediction-driven method. [13]–[15] employ resource demand prediction to achieve elastic resource allocation without assuming prior knowledge of the applications in the cloud. [15] is slightly different, as it uses VM replication to reduce application start-up times.

A few works consider coordinated VC scaling on both optimization of the VCs’ localities and the elastic resource allocation. [16] proposes a system that allows tenants to dynamically request and update minimum guarantees for both network bandwidths and compute resources at runtime; this is realized using the resource reservation method. [17] studies survivable and bandwidth-guaranteed embedding of virtual clusters, and it proposes a novel algorithm to jointly optimize primary and backup embeddings of the virtual clusters. These works on VC resource allocation fail to fully consider both localities and elasticities for the scaling requests in one determined time period. In this paper, we jointly consider VC placement on localities and elasticities with scaling fluctuation to maximize the over-time elasticity during one time period with minimal extra cost in DCNs.

III. MODEL AND PROBLEM FORMULATION

A. PLATFORM MODEL

This paper focuses on the elastic VC scaling placement problem for the hose communication model in fat-tree. We jointly consider localities and elasticities during resource allocation for VC scaling, and we use elasticity to measure the growth potential for VCs, an important factor for weighting flexibility during the placement. Our objective is to maximize the elasticity for VCs with a communication cost constraint in the DCN.

B. DATA CENTER MODEL

In this paper, we consider the fat-tree as our data center network topology model. Fat-tree is an extended tree topology which has been applied to DCNs by several researchers [18]. In fat-tree, each \( \theta \)-port switch in the edge layer is connected to \( \frac{\theta}{2} \) PMs [18]. Each PM in the fat-tree is denoted as \( M_i \) and divided into multiple slots where VMs can be placed. The
capacity of each PM is denoted by $C_i$, and the PMs in the DCN are homogeneous. The PLs are denoted by $L = \{L_{ij}\}$, and the capacity of PL is denoted by $B_{ij}$. $T_{S_{ij}}$ denotes the subtree under the root (physical switch) $S_{ij}$ that contains a set of PMs and PLs. In this paper, we set the root (physical switch) $S_{ij}$ as the locality, which we use to denote the position of the virtual cluster $V_i$. There are two properties of the locality, private and public. When the property is private, the resource of the subtree $T_{S_{ij}}$ can only be used by $V_i$. Otherwise, the resource under the subtree $T_{S_{ij}}$ can be used by any VC.

C. VIRTUAL CLUSTER (VC)

The VC is an abstraction that allows each tenant to specify both the virtual machines (VMs) and per-VM bandwidth demand of its service [19]. Let $V_i$ denote the $i^{th}$ existing VC in the DCN. Each VC consists of a set of VMs and one virtual switch where $V_i = (< N_i, B_i>)$. $N_i$ is the number of VMs in the $i^{th}$ VC, and $B_i$ is the bandwidth demand between VMs and the virtual switch. In this paper, we consider the hose model based on the VC abstraction. In the hose model, each customer specifies a set of endpoints to be connected with a common endpoint-to-endpoint performance guarantee [20].

1) Communication Cost

Since a good locality means that the allocation of a virtual cluster to reduce the communication latency among VMs, we define a new function can measure communication cost. The standard metric to evaluate communication cost is measuring the embedded footprint [1], [21], [22]. During the VM placement, we try to minimize the communication cost. For each virtual request $V_i$,

$$m(V_i) := \sum_{j=1}^{3} |T_{S_{ij}}| \cdot H_j \cdot \gamma$$

$|T_{S_{ij}}|$ denotes the total amount of VMs under the subtree $S_{ij}$ of virtual cluster $V_i$, as shown in equation (1). $H_j$ is the hops between PMs holding the VMs of $V_i$. Since the architecture of the DCN is fat-tree in this paper, the value of $H_j$ is $H_1 = 2$, $H_2 = 4$, $H_3 = 6$. $\gamma$ is a constant value which denotes the communication cost between each pair of VMs in $V_i$. The communication cost of a virtual request can be calculated via the following case distinction: (1). If all VMs of $V_i$ place into one PM, the communication cost $m(V_i) = 0$. (2). If $V_i$ places under the ToR switches or aggregation switches of a pod, the communication cost $m(V_i) = 2 \cdot |S_{L_{i1}}| + 4 \cdot |S_{L_{i1}}|$. (3). If $V_i$ places under the core switches of different pods, the communication cost $m(V_i) = 6 \cdot |S_{L_{i1}}|$. 

2) Elasticity

Let $E_i$ denote the elasticity of $V_i$, which measures the growth potential of $V_i$ under the communication cost constraint. In this paper, we use this factor to weigh the flexibility of the placement of the VCs.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_i$</td>
<td>PM in the DCN</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Capacity of the $i^{th}$ PM in the DCN</td>
</tr>
<tr>
<td>$L_{ij}$</td>
<td>PL in the DCN</td>
</tr>
<tr>
<td>$B_{ij}$</td>
<td>Capacity of PL in the DCN</td>
</tr>
<tr>
<td>$V_i$</td>
<td>The $i^{th}$ existing VC in the DCN</td>
</tr>
<tr>
<td>$N_i$</td>
<td>Existing VMs of $V_i$</td>
</tr>
<tr>
<td>$N'_i$</td>
<td>Scaling VMs of $V_i$</td>
</tr>
<tr>
<td>$B_i$</td>
<td>Existing bandwidth demand of $V_i$</td>
</tr>
<tr>
<td>$\delta B_i$</td>
<td>Scaling bandwidth demand of $V_i$</td>
</tr>
<tr>
<td>$T_{S_{ij}}$</td>
<td>Subtree of $V_i$ under the locality (root) $S_{ij}$</td>
</tr>
<tr>
<td>$R^M_{S_{ij}}$</td>
<td>Available computing resource in the subtree $T_{S_{ij}}$ for $V_i$</td>
</tr>
<tr>
<td>$R^L_{S_{ij}}$</td>
<td>Available communication resource in the subtree $T_{S_{ij}}$ for $V_i$</td>
</tr>
<tr>
<td>$m(\cdot)$</td>
<td>Communication cost of $V_i$</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>Upper bound for communication cost</td>
</tr>
<tr>
<td>$\rho_i$</td>
<td>Scaling ratio for $V_i$</td>
</tr>
<tr>
<td>$\zeta_i$</td>
<td>Adjust factor of the elasticity for $V_i$</td>
</tr>
</tbody>
</table>

Definition 1: Combinational elasticity is defined as $E_i = \min\{E_i^M, E_i^L\}$, where $E_i^M$ is defined as the minimum percentage of available VM slots among PMs under the subtree $T_{S_{ij}}$ of $V_i$.

$$E_i^M = \min_{c_i \in T_{S_{ij}}} \{1 - \frac{\max_i (C_i^* + C_i')}{C_i} \} \quad (2)$$

Similarly, $E_i^L$ is defined as the minimum percentage of available communication resource (bandwidth) among PLs under the subtree $T_{S_{ij}}$ of $V_i$, in equation (3).

$$E_i^L = 1 - \frac{f(\sum_{c_i \in T_{S_{ij}}} (C_i^* + C_i'))}{B_{ij}} \quad (3)$$

$C_i^*$ and $C_i'$ denote the number of existing and incoming VMs of $V_i$ belong to the PMs under the subtree $T_{S_{ij}}$. Similarly, $f(\sum_{c_i \in T_{S_{ij}}} (C_i^* + C_i'))$ denotes the communication demand under the subtree $T_{S_{ij}}$, where $f(\cdot)$ denotes the bandwidth demand function of VM communications.

D. BASIC PROBLEM FORMULATION

1) Definition of VC Scaling Placement

In this paper, we consider the VC placement based on the hose model and fat-tree. Let $V_i = (< N_i, B_i>)$ denote the $i^{th}$ VC, which contains several types of problem instances for scaling up. The basic ones are either to increase the cluster size on the computing resource from $N_i$ to $N_i + N_i'$ or to increase the communication resource from $B_i$ to $\delta B_i$. The most difficult is to increase both computation and communication resources, which range from $N_i$ to $N_i + N_i'$ and from $B_i$ to $\delta B_i$, i.e., $V_i = (< N_i, B_i>)$ or $V_i = (< N_i + N_i', \delta B_i>)$. We mainly focus on the latter case, combined resource scaling , and the algorithms we propose for it can also efficiently solve the other two types of scaling problems.

2) Objective Function

Our objective is to maximize the elasticity of $V_i$ with the constraint of communication cost. We use a constant $\Phi$ to denote
the upper bound of the communication cost that is initialized by the users based on the demands of the applications [21]. Our problem can be formally formulated as follows:

\[
\text{maximize } E_i \\
\text{subject to } 0 \leq m(V_i) \leq \Phi_i \\
C^*_i + C'_i \leq C_i \\
f(\sum_{C_i \in P_{S_{ij}}} (C^*_i + C'_i)) \leq B_{ij}
\]

Variables are \(C^*_i\) and \(B'_{ij}\), and \(E_i\) is derived. Others are given, i.e., \(C'_i\), \(B'_i\), and \(\Phi_i\). Equation (5) shows the constraints on the communication cost \(m(V_i)\) where the value should be greater than or equal to 0 with the upper bound \(\Phi_i\). Equation (6) shows the constraints on capacities of PMs, in which the total number of incoming and existing VMs \(C^*_i\) and \(C'_i\) on the \(i\)th PM cannot exceed the capacity \(C_i\). Equation (5) is the link capacity constraint, which shows that the bandwidth usage of both the existing and incoming VMs on \(L_{ij}\) under the subtree \(T_{S_{ij}}\) can not exceed the link capacity \(B_{ij}\). The major notations used in this paper are listed in Table I.

IV. SINGLE VIRTUAL CLUSTER SCALING

This section proposes one optimal solution, VCS, that can be applied to deal with the scaling of a single virtual cluster.

A. ALGORITHM AND DESCRIPTION

1) Initialization

We take the incoming scaling request for \(V_i\) with \((N'_i, \delta B_i)\) at time slot \(t_i\) as the input, and the output is the occupation state for \(V_i\) in the DCN. The initialization in line 1 is to find the locality \(S_{ij}\) of \(V_i\) and calculate the available physical resource under the subtree \(T_{S_{ij}}\). The available resource contains both computation resource \(R^M_{S_{ij}} = \sum_{M_i \in T_{S_{ij}}} C^*_i\) and communication resource \(R^L_{S_{ij}} = \sum_{L_{ij} \in T_{S_{ij}}} B'_{ij}\). We also initialize the upper bound of the communication cost \(\Phi_i\) for \(V_i\), which users usually set as the QoS guarantee.

2) Virtual Cluster Scaling (VCS)

For each scaling request, we try to find the appropriate subtree to obtain enough physical resources. In lines 2 and 3, we first compare the incoming scaling request \((N'_i, \delta B_i)\) with the total available physical resource, \(R^M_{S_{ij}}\) and \(R^L_{S_{ij}}\), under \(T_{S_{ij}}\). If the total amount of available physical resource cannot satisfy the scaling request, the current subtree root will be positively adjusted by the step \(T_{S_{ij}} \leftarrow T_{S_{ij}+1}\) in line 5. This process ends when the locality moves to the upper bound \(S'_{ij}\). We start to place scaling requests in line 6 using the function \(VMP(N'_i, \delta B_i)\), which is described in Algorithm 2.

3) VM Placement (VMP)

In this section, we propose an efficient algorithm to find a valid allocation for the scaled virtual cluster with optimization of the VMs’ localities. The initialization in line 1 calculates the partial elasticity under the subtree \(T_{S_{ij}}\), which is denoted as \(E_{T_{S_{ij}}}\). To adjust the partial elasticity under \(T_{S_{ij}}\), we define a factor \(\zeta_i\). Before allocating the computation resource, we check the scaling condition of the communication resource in line 2, which can ensure that the bandwidth \(B_i\) for each VM appropriately reserves communication resource on physical links. If \(\delta\) is not equal to 1, the communication resource has to scale or release, and we update the bandwidth capacities for the existing VMs of \(V_i\) under the communication demand \(\delta B_i\) in line 3. Otherwise, the communication resource does not have any scaling or releasing. After that, we compute the available computation and communication capacities under the subtree \(T_{S_{ij}}\) with the limitation of \(E_{T_{S_{ij}}}\) in line 4. From lines 5 to line 9, we start to allocate computation resource under the subtree \(T_{S_{ij}}\), and we place \(N'_i\) VMs into PMs based proportionally on the remaining available capacities. If the total number of available physical resources cannot satisfy the scaling request, the current partial elasticity is negatively adjusted by the step factor \(\zeta_i\), as shown in lines 8 and 9. This process ends when the partial elasticity \(E_{T_{S_{ij}}}\) is 0. In line 10, the communication demands of the placed VMs are evenly split into paths that connect them under the subtree \(T_{S_{ij}}\).

B. OPTIMALITY ANALYSIS

Theorem 1: VCS is an optimal solution for the \(V_i\) placement under the communication cost constraint \(\Phi_i\).

Proof: The maximum number of servers in a fat-tree is \(\frac{n^2}{4}\). We start to prove from \(\theta = 2\), which contains 2 PMs in fat-tree. For the virtual cluster \(V_i\), we suppose that VCS is not an optimal solution, meaning that another solution \(O\) will be the optimal one. Since the bandwidth resource is not oversubscribed, the bottleneck of the elasticity exists on the computation resource. Let \(C_a\) and \(C_b\) denote the remaining available slots of PMs \(a\) and \(b\) under the subtree \(T_{S_{ij}}\) of \(V_i\). In order to place the scaling \(N'_i\) VMs, we assume the optimal solution \(O\) is \((x, y) \ (x + y = N')\), where \(x\) VMs are placed on \(a\) and \(y\) VMs are placed on \(b\). Similarly, we suppose the solution calculated by VCS is \((u, v) \ (u + v = N')\), where \(u\) VMs are placed on \(a\) and \(v\) VMs are placed on \(b\). We suppose
Algorithm 2 VM Placement (VMP)

Input: Scaling request \( V_i \) with \( \{N_i^t, \delta B_i\} \);
Output: DCN occupation state for \( V_i \);
1: Initialize \( E_i \) and adjust factor \( \zeta_i \) for \( V_i \);
2: if \( \delta \neq 1 \) then
3: Update the capacities of PLs for existing VMs of \( V_i \) according to the scaling \( B_i \rightarrow \delta B_i \);
4: end if
5: Compute \( R^T_{E_i} \) and \( R^M_{E_i} \) according to \( E_i \) under \( S_{ij} \);
6: while \( E_i > 0 \) do
7: if \( N_i^t \leq R^T_{E_i} \) then
8: Place \( N_i^t \) VMs into PMs in \( S_{ij} \) (proportion based on the remaining available capacities of PMs);
9: else if \( N_i^t > R^T_{E_i} \) then
10: Update \( E_i = E_i - \zeta_i \);
11: end if
12: end while
13: Communication demands of placed VMs are evenly split into paths connecting them;

for the multiple \( \varpi \) VCs may lead to different results. Take Fig. 3 for example; three VCs exist in the DCN, and the numbers of existing VMs for these users are 5, 2, and 7, respectively. The communication costs for the three VCs have already changed into the position of subtree root, which is marked by cycles with different corresponding colors in Fig. 3. When all three VCs send their scaling requests, we schedule them in the order \( VCl \rightarrow VC2 \rightarrow VC3 \), as shown in Fig. 3 (a). Then, we have the elasticities \( VCl = \frac{1}{5} \), \( VC2 = \frac{1}{7} \), and \( VC3 = \frac{1}{9} \), respectively. The over-time elasticity for all three VCs is \( \frac{1}{5} + \frac{1}{7} + \frac{1}{9} = \frac{7}{15} \). When we change the scheduling order to \( VC2 \rightarrow VC3 \rightarrow VCl \), as shown in Fig. 3 (b), the elasticities are \( VC1 = \frac{1}{5} \), \( VC2 = \frac{1}{7} \), and \( VC3 = \frac{2}{9} \), respectively. The over-time elasticity for all three VCs under this scheduling strategy is \( \frac{1}{5} + \frac{1}{7} + \frac{2}{9} = \frac{7}{15} \).

Theorem 2: The MVCS placement for the over-time elasticity maximization problem is NP-hard.

Proof: Given a set of scaling VCs, \( V = \{V_1, V_2, ..., V_{\varpi}\} \). Let the amount of existing VCs be \( \varpi \), and they request to scale at the same time \( t \). We assume the rest of the available resource of the DCN at \( t \) is \( R \). The communication cost of each VC is related to the locality of its placement, which has a limitation \( \Phi \) defined by the uses. The goal is to place all \( \varpi \) VCs using the fewest physical resources with determined capacities under the communication cost \( \Phi \). We reduce the original problem to the variable-sized bin-packing problem [22], [23], an NP-hard problem that finds an assignment that uses the fewest bins. Thus, the MVCS placement for the over-time elasticity maximization problem is NP-hard.

Before describing the algorithm, we first introduce an important parameter scaling ratio.

Definition 3: Let \( \rho_i \) denote the scaling ratio of the virtual cluster \( i \), which is the ratio between the scaling amount of \( V_i \) and the maximum available physical resource \( R_{S_{ij}} \) under the subtree \( T_{S_{ij}} \),

\[
\rho_i = \frac{V_i}{R_{S_{ij}}} \quad (8)
\]

We use \( \rho_i \) measure the scaling level of the virtual cluster \( V_i \). Since the physical resource \( R_{S_{ij}} \) for \( V_i \) is limited by the communication cost \( \Phi_i \), the scaling amount of \( V_i \) should be under the constraint in Equation (8). Therefore, the range of the scaling ratio is \( 0 < \rho_i < 1 \).

B. ALGORITHM

Since MVCS placement is an NP-complete problem, we propose a heuristic algorithm to find a consistent scaling scheduling order that improves the over-time elasticity as much as possible. We take the incoming scaling request set \( V = \{V_1, V_2, ..., V_{\varpi}\} \) as the input, and the output is the occupation state for \( V \) in the DCN.

The initialization in line 1 finds the localities \( S_{ij} \) for VCs and calculates the available physical resource under the subtree \( T_{S_{ij}} \). Based on that, we initialize the scaling ratio \( \rho_i \), which is the ratio between the scaling amount of \( V_i \) and the available physical resource under the subtree \( T_{S_{ij}} \), i.e.
VI. ONLINE MULTIPLE VIRTUAL CLUSTER SCALING

In this section, we describe the online condition for the multiple VC scaling problem with synchronous and asynchronous incoming rates.

A. SYNCHRONOUS ONLINE MULTIPLE VIRTUAL CLUSTER SCALING (S-OMVCS)

In this section, we address the online multiple virtual cluster scaling problem with a synchronous incoming rate. The scaling requests of virtual clusters in set \( V \) are incoming at the same time slot and also release at the same time slot. Multiple VCs can make scaling requests simultaneously, but a single time slot can only deal with one VC scaling request. We consider the performance of VCs in a single time period \([0, T]\) using over-time elasticity. For each virtual cluster, the incoming scaling amount is uncertain, and we need to preprocess existing virtual clusters, including priority ranking and future prediction. The future prediction is based on the Bayesian parameter estimation, as discussed below. Let \( \rho_i^* \) denote the maximum scaling ratio for the virtual cluster \( V_i \), which means that the flexibility of resources under the subtree \( S'_{ij} \) should not exceed the communication cost \( \Phi_i \). The priority ranking for multiple scaling requests is the same as in offline requests, which depend on both the upper bound of the root position \( S_{ij} \) and the scaling ratio \( \rho_i \).

1) Bayesian Parameter Estimation for S-OMVCS

Based on the current incoming scaling \( V_i \) request, we use Bayesian parameter estimation [24] to predict the future fluctuation statement for each virtual cluster. We use the historical fluctuation statement before \( T_n \) as our sample, denoted as \( \mathbb{I} = \{N_i^t|_{t=0,T_n}\} \). Let \( n \) denote the number of samples in \( \mathbb{I} \), which means that the current location is at the \( n + 1 \) time slot. As we move through the time slots, the new observation samples are obtained, and the posterior probability density function is sharpened to form the largest spike near the true value of the parameter. We use Bayesian parameter estimation based on uniform distribution to predict the future information of the scaling virtual cluster \( V_i \), and each virtual cluster is independent and identically distributed. We use the standard Gaussian distribution as the prior distribution, i.e., \( \mu_0 = 0 \) and \( \delta_0^2 = 1 \); this is the same as in [26]. Before doing the prediction, we first calculate the maximum likelihood \( \mu' \) for the sample \( \mathbb{I} \), where \( \mu' = \frac{1}{n} \sum_{t=1}^{n} N_i^t \). According to the prior and maximum likelihood values, we have the posterior distributions \( \mu = \frac{n}{n+\delta^2} \mu' \) and \( \delta^2 = \frac{\delta^2}{n+\delta^2} \). Based on the Bayesian estimation, we have the probability

![Algorithm 3 Multiple Virtual Cluster Scaling (MVCS)](image)

**Algorithm 3 Multiple Virtual Cluster Scaling (MVCS)**

**Input:** Scaling set \( V = \{V_1, V_2, \ldots, V_\omega\} \);  
**Output:** DCN occupation state for \( V \):

1. Initialize the localities \( S_{ij} \) and \( S'_{ij} \), \( \rho_i \), and \( \Phi_i \) for VCs;  
2. Sort VCs in the set \( V \) to \( V' \) by localities \( i = \arg \min_i S'_{ij} \);  
3. For VCs with the same localities, prioritize by scaling ratio \( i = \arg \min_i \rho_i \);  
4. for \( i = 1 \) to \( i = \omega \) in \( V' \) do  
5. Place \( V_i \) into the DCN;  
6. Same as Algorithm 1 form line 2 to line 5;  
7. end for

\( 0 < \rho_i < 1 \). After that, we initialize the upper bound of the root position \( S'_{ij} \) based on the communication cost \( \Phi_i \) for VCs in the set \( V \). In line 2, we first sort VCs in the set \( V \) to \( V' \) by localities \( i = \arg \min_i S'_{ij} \) based on the communication cost \( \Phi_i \). Let \( V' \) be the sorting result of the scaling VCs. If the level of \( S'_{ij} \) is the same for the VCs, let the lower scaling ratio \( \rho_i \) have higher priority in line 3. From lines 4 to 6, we start to place the VCs in the DCN by prioritizing the VCs in the set \( V' \) with the lower localities and scaling ratios \( \rho_i \). The placement process for each VC is the same as in Algorithm 1, line 6.
Algorithm 4 Synchronize Online Multiple Virtual Cluster Scaling (S-OMVCS)

Input: Scaling set \( V = \{V_1, V_2, ..., V_N\} \) at time slot \( t_i \);
Output: DCN occupation state for \( V \);
1: Initialize the localities \( S_{ij} \) and \( S'_{ij} \), \( \rho_i \) and \( \Phi_i \) for VCs;
2: for \( i = 1 \) to \( i = \infty \) in \( V \) do
3: Estimate the fluctuating mean \( \mu_i \) for \( V_i \) based on bayesian parameter estimation;
4: Calculate the future scaling ratio \( \rho_i^* \) based on the \( \mu_i \);
5: Relocate the locality property \( S'_{ij} \) based on the \( \rho_i^* \);
6: Make resource reservations according to \( S'_{ij} \) for \( V_i \);
7: end for
8: Sort VCs in the set \( V \) to \( V' \) by localities \( i = \arg \min S'_{ij} \);
9: for \( i = 1 \) to \( i = \infty \) in \( V' \) do
10: For VCs with the same localities, prioritize by scaling ratio \( i = \arg \min \rho_i^* \);
11: Place the scaling request \( V_i \) into the DCN;
12: Same as Algorithm 1 form line 2 to line 8;
end for

density function in Equation (9).

\[
p(\mu_i | I) = \frac{1}{(\sqrt{2\pi})^d} \exp \left[-\frac{1}{2\delta^2} \sum_{i=1}^{n} (\mu_i - \mu)^2 \right] \tag{9}
\]

Algorithm and Description

We take the set of scaling requests \( V \) arriving at time slot \( t_i \) as the input, and the output is the occupation state for \( V \) in the DCN. Since the scaling requests of the virtual clusters in set \( V \) are synchronously incoming, we use the same initialization as MVCS (Algorithm 3, in line 1). In lines 2 to 7, we start to estimate the future information for the incoming virtual clusters. For each virtual cluster, we first estimate the fluctuating amount of \( \mu_i \) and \( \delta \) for \( V_i \) based on Bayesian parameter estimation. Then, we calculate the future scaling ratio \( \rho_i^* \) based on the \( \mu_i \), and we relocate the locality property \( S'_{ij} \) based on the \( \rho_i^* \). We use \( S'_{ij} \) for resource reservation for \( V_i \). In line 8, we sort VCs in the set \( V \) to \( V' \) by localities \( i = \arg \min S'_{ij} \) and start to deploy each VC in \( V' \). From lines 9 to 13, we start to deploy each VC in \( V' \) one-by-one based on the sorting order. VCs with the same localities are prioritized by the scaling ratio \( i = \arg \min \rho_i^* \). In line 11, we start to place the \( V_i \) into the DCN, and the placement process for each \( V_i \) is the same as Algorithm 1 from lines 2 to 8.

B. ASYNCHRONOUS ONLINE MULTIPLE VIRTUAL CLUSTER SCALING (A-OMVCS)

In this section, we address the online multiple virtual cluster scaling problem with an asynchronous incoming rate. In this condition, the scaling requests of virtual clusters in set \( V \) are incoming at different time slots and also release at different time slots. For each time slot, the incoming scaling amount of virtual clusters is uncertain. Let the fluctuation of incoming numbers of the VCs with Gaussian distribution be \( N(\mu, \delta^2) \). Since the scaling amount of the VCs is uncertain at each time slot, if we use the same reservation scheme in Algorithm 4, resource utilization will be inefficient. Therefore, we need to predict not only the scaling information for each virtual cluster, but also the incoming amount of virtual clusters at the next time slot. We preprocess each currently existing virtual cluster in the same way as in the synchronous case. In the asynchronous case, multiple VCs can make scaling requests together at the same time slot, but each time slot can only deal with one VC scaling request. We also consider the performance of VCs in a single time period \([0, T] \) using over-time elasticity.

1) Bayesian Parameter Estimation for A-OMVCS

In the asynchronous online scaling case, we first do the prediction for the incoming scaling VCs. We use Bayesian parameter estimation to predict the future fluctuation statement for virtual clusters. Let \( \kappa_i \) denote the incoming VCs at time slot \( t_i \); we use the historical fluctuation statement \( Z = \{\kappa_i | \in [0, t_n]\} \) of VCs before \( T_n \) as our sample. Let \( n \) denote the number of samples in \( Z \), which means the current location is at the \( n + 1 \) time slot. We use the standard Gaussian distribution as the prior distribution, i.e., \( \hat{\mu}_0 = 0 \) and \( \delta_0^2 = 1 \). Then, we calculate the maximum likelihood \( \hat{\mu}^* \) for the sample \( Z \), where \( \hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \kappa_i \). According to the prior and maximum likelihood values, we have posterior distributions where \( \hat{\mu} = \frac{n}{n + \delta^2} \hat{\mu}^* \) and \( \delta^2 = \frac{n}{n + \delta^2} \delta^2 \). Based on the Bayesian estimation, we have the probability density function in Equation (10).

\[
p(\hat{\mu} | Z) = \frac{1}{(\sqrt{2\pi})^d} \exp \left[-\frac{1}{2\delta^2} \sum_{i=1}^{n} (\hat{\mu}_i - \hat{\mu})^2 \right] \tag{10}
\]

Algorithm and Description

We use the same input and output settings as Algorithm 5, which includes \( \kappa_i \) VCs at time slot \( t_i \). In line 1, we do the same initialization as MVCS (Algorithm 3). Then, we estimate the fluctuating mean \( \hat{\mu}_i \) for incoming VCs based on Bayesian parameter estimation in line 2. According to the mean \( \hat{\mu}_i \), we calculate the scaling amount \( \hat{\mu}_i \) of VCs in line 3. In lines 4 to 7, we start to estimate future information for each of the incoming virtual clusters, and we sort VCs in the set \( V \) to \( V' \) by localities \( i = \arg \min S'_{ij} \). In lines 8 to 10, we make resource reservations for the \( \kappa \) VCs in the set \( V' \) that has the highest probability for scaling requests in the incoming time slots. In lines 11 to 13, we start to place the \( \kappa \) VCs into the DCN, and the placement process is the same as in Algorithm 4 from lines 9 to 13.

VII. EXPERIMENTS

This section conducts extensive simulations to study elastic VC scaling placement under three aspects: single VC scaling, multiple VC scaling, and online multiple VC scaling. These experiments are conducted to evaluate the performances of the proposed algorithms. After presenting the datasets and
Algorithm 5 Asynchronous Online Multiple Virtual Cluster Scaling (A-OMVCS)

Input: Scaling set $V = \{V_1, V_2, ..., V_m\}$ at time slot $t_i$;

Output: DCN occupation state for $V$;

1: Initialize the localities $S_{ij}$ and $S'_{ij}$, $\rho_i$ and $\Phi_i$ for VCs at time slot $t_i$;
2: Estimate the fluctuating mean $\hat{\mu}_i$ for incoming VCs based on Bayesian parameter estimation;
3: Calculate the scaling amount $\hat{\kappa}_i$ of VCs based on the $\hat{\mu}_i$;
4: for $i = 1$ to $i = k$ in $V$ do
5:   Same as Algorithm 4 from line 3 to line 5;
6: end for
7: Sort VCs in the set $V$ to $V'$ by localities $i = \arg\min_i S'_{ij}$;
8: for $i = 1$ to $i = k$ in $V'$ do
9:   Make resource reservations according to $S'_{ij}$ for $V_i$;
10: end for
11: for $i = 1$ to $i = k$ in $V'$ do
12:   Same as Algorithm 4 from line 9 to line 13;
13: end for

settings, the results are shown from different perspectives to provide insightful conclusions.

A. SINGLE VIRTUAL CLUSTER SCALING

1) Experiment Setting

The DCN is modeled as a fat-tree, in which the number of switches’ ports are $\theta = 4$, $\theta = 6$, and $\theta = 8$. Let the amount of PMs in the fat-tree be fully connected with the maximum numbers, which are 16, 54, and 128, respectively. The supplied computing and communication resources of the PMs and PLs are real numbers uniformly distributed between 50 and 100 units. For each group with a different switch’s port, we calculate elasticity after the scaling placement process. The results are averaged 10 times for each algorithm. We compare the proposed VCS algorithm with the two benchmark algorithms in a number of trace-driven settings.

- Equally Scaling (ES): the scaling request of $V_i$ is evenly divided into several pieces depending on the amount of PMs in the sub-tree. It can obtain the load-balance for each virtual request [1].
- Greedy Scaling (GS): the scaling request of $V_i$ for the PMs depends on the amount of available resource in the sub-tree; PMs with high margins have a high priority.

2) Experiment Results

Fig. 4, Fig. 5, and Fig. 6 present the elasticity for the single VC scaling condition, in which the numbers of the switches’ ports are $\theta = 4$, $\theta = 6$, and $\theta = 8$, respectively. For each group experiment, we use the same three algorithms: ES, GS, and VCS, and we calculate averaged 10 times of the elasticities for various scaling requests. Additionally, we have the following observations: (i) The elasticity of the scaling VC depends on the architecture of the fat-tree. Since the construction of DCNs is based on the number of switches’ ports, we can see that the elasticity of the VC with scaling
under the $\theta = 4$ switch is much lower than that of the fat-trees where $\theta = 6$ and $\theta = 8$. (ii) The elasticity of the scaling VC depends on various placement algorithms. As shown in Fig. 4, Fig. 5, and Fig. 6, the performance of GS decreases significantly with the increase in the scaling of VC. ES’s performance depends on the existing localities of the existing VMs. Therefore, the fluctuation of FFRP is much larger in ES than in other algorithms. Compared with ES and GS, VCS has the best performance in elasticity.

B. MULTIPLE VIRTUAL CLUSTER SCALING

1) Experiment Setting

This section evaluates the elasticity of the scaling of multiple VCs scaling and uses the same data set as the single VC scaling problem. Set the VMs of the VCs scaled at one time slot are evenly distributed between 0 and 50, the bandwidth demands $\delta$ scale between 0 and 1. Let the switch’s port be $\theta = 4$, $\theta = 6$, and $\theta = 8$ for each group. In addition to the proposed algorithms, three baseline algorithms are used:

- Random Schedule Scaling (RSS): the scheduling order for the multiple VCs is random.
- Decreasing Schedule Scaling (DSS): the scheduling order for the multiple VCs is decreasing.
- Increasing Schedule Scaling (ISS): the scheduling order for the multiple VCs is increasing.

2) Experiment Results

Since the scale of the VMs ranges from 0 to 50, we calculate the even value of the over-time elasticity under different $\delta$ between 0 and 1. We use the over-time elasticity to evaluate the performance of the proposed algorithm, and we compare it with three base-line algorithms: RSS, DSS, and ISS. Fig. 7 presents the over-time elasticity of the scaling of multiple VCs using different schedule strategies. The number of scaling VCs at the time slot is evenly distributed from 0 to 50. For each time slot, we allow one VC to be processed. According to the simulation results, we have the following observations: (i). The volatility of the multiple scaling VCs is stable. As shown in Fig. 7, the mean values are marked by red lines, which are close to each other under different algorithms. (ii). The over-time elasticity for the multiple VCs depends on the scheduling order. Comparing these four algorithms, the performance of RSS is the worst. The interval range of ISS is better than that of DSS, which depends on the distribution of existing VMs of the VCs. Compared with RSS, DSS, and ISS, MVCS has the best performance in over-time elasticity.

C. ONLINE MULTIPLE VIRTUAL CLUSTER SCALING

1) Experiment Setting

This section evaluates the elasticity of online multiple VC scaling, in which the arrival times of VCs are discretionary and scaling amounts are randomly determined by tenants. We divide the scaling into two parts: synchronous online multiple VC scaling and asynchronous online multiple VC scaling. We set the scaling frequency of the VCs to 1, which means that each time slot has to process the scaling or releasing requests. We run each of our simulations for 10 time slot intervals. The parameters and symbols that we vary in our simulations are over-time elasticity.
We propose heuristic algorithms MVCS, S-OMVCS, and A-OMVCS for the over-time elasticity maximization problem. This paper focuses on the condition that VCs scaling on both computation and communication resources, which can also be adapted to each individual resource. Extensive simulations demonstrate that our elastic VC scaling placement schemes outperform existing state-of-the-art methods in the DCN in terms of elasticity.

**ACKNOWLEDGMENT**

The work of the first author was done during her stay as a visitor scholar at Temple University. This research was supported in part by NSF grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, CSC 20163100, and CNS 1439672.

**REFERENCES**


**VIII. CONCLUSION**

This paper considers elastic scaling for existing VCs to maximize elasticity in the DCN with the constraint of communication cost. We achieve this using a resource allocation scheme, VCS, which comes with provable optimality guarantees for single VC scaling. After that, we extend our scheme to scale for multiple VCs, and we prove that scaling multiple VCs for the over-time elasticity maximization problem is NP-hard. We propose heuristic algorithms MVCS, S-OMVCS, and A-OMVCS for both offline and online conditions in the multiple VC scaling problem.


Shuaibing Lu received her M.S in computer science and technology from Jilin University, Changchun, China, and she is currently a Ph.D. student. She was a researcher at the distributed computing system and data center network. She is supported by the China Scholarship Council as a visiting scholar supervised by Prof. Jié Wu in the department of computer and information sciences at Temple University (2016-2018). She is a student member of the IEEE.

Zhiyi Fang received his PhD in computer science from Jilin University, Changchun, China, in 1998. He currently works there as a professor of computer science. He was a senior visiting scholar at the University of Queensland, Australia, 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.

Jie Wu is currently the chair and a Laura H. Carnell professor at the Department of Computer and Information Sciences, Temple University. Prior to joining Temple University, he was a program director at the National Science Foundation and a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He has served on several editorial boards, including IEEE Transactions on Computers, IEEE Transactions on Service Computing, and the Journal of Parallel and Distributed Computing. He was the general co-chair/chair for IEEE MASS 2006 and IEEE IPDPS 2008 and the program co-chair for IEEE INFOCOM 2011. He is currently the general chair for IEEE ICDCS 2013 and ACM MobiHoc 2014, and the program chair for CCF CNCC 2013. He was an IEEE Computer Society distinguished visitor, ACM distinguished speaker, and the chair for the IEEE Technical Committee on Distributed Processing. He is a China Computer Federation (CCF) distinguished speaker and a fellow of the IEEE. He received the 2011 China Computer Federation Overseas Outstanding Achievement Award.

Guannan Qu received her B.E. in computer science and technology from Jilin University, Guangzhou, China in 2003, and her M.S and PhD degrees in computer science from Jilin University, Changchun, China in 2007 and 2010, respectively. She is currently a lecturer of computer science and technology at Jilin University. She was financially supported by the China Scholarship Council as a visiting scholar at the University of Texas at Dallas from 2008 to 2010. Her research interests include computer system architecture, performance analysis, and quality-of service issues in high-speed networks. She is a member of the IEEE.

* * *