Towards Dynamic Request Updating with Elastic Scheduling for Multi-tenant Cloud-based Data Center Network

Shuaibing Lu, *Member, IEEE,* Jie Wu, *Fellow, IEEE,* Jiamei Shi, Juan Fang, *Member, IEEE,* Jiayue Zhang, and Haiming Liu

Abstract—Serving the ever-growing demand for computation, storage, and networking resources for multi-tenant in cloud computing is an important mission of Data Center Networks (DCNs). In this paper, we study the dynamic request updating problem, and our objective is to maximize the elasticity of cloud-based DCNs while achieving rapid response to multi-tenants. We use virtual clusters under the hose communication model to denote requests. Instead of using heuristic algorithms as the existing work does, this paper introduces a novel two-stage dynamic request updating framework with elastic resource scheduling strategy. In the first stage, we propose a multi-tenant fast initial provisioning scheme to realize the real-time response and analyze its optimality and complexity. Additionally, we provide a deep reinforcement learning-based dynamic updating strategy to enhance the elasticity of virtual clusters that are being used or scaling during the second stage. We train a fully connected neural network by creating a new feasible action set to realize the reduction, and it approximates the policy based on a proposed aggressive objective selection method to improve training speed while avoiding high dimensions caused by large scales of tenants and DCNs. Extensive evaluations demonstrate that our scheme outperforms baselines in terms of both elasticity and efficiency.

Index Terms—data center network, dynamic request updating, multi-tenant, elastic scheduling, resource provisioning

✦

1 INTRODUCTION

W ITH the ever-increasing demand of cloud services, the data center network (DCN) has become an efficient and promising data processing infrastructure for cloud computing. As reported in the public data of Microsoft Azure [1], the demand and deployment size of the tenant is very bursty and unpredictable in terms of memory, cores, and bandwidth. One fundamental challenge of DCNs is to serve the varying needs of multi-tenant without requiring frequent provisioning changes. This paper proposes an elastic resource provisioning scheme to deal with the scaling without load redistribution during a run time. In order to simplify the description of the resource provisioning problem, we use virtual clusters to denote the requests of multi-tenant, and each virtual cluster is an abstraction of a set of virtual machines (VMs), which has the requirements on both computing and communication resources [3]. We use the notion of elasticity to measure the potential growth of multi-tenant in terms of computing and communication resources at the same time, which is defined as the degree of a system that is able to adapt to the workload changes by provisioning and releasing resources in an autonomic manner [5], [6], [8]. We consider the virtual clusters with hose model under constraints, and our objective is to maxi-
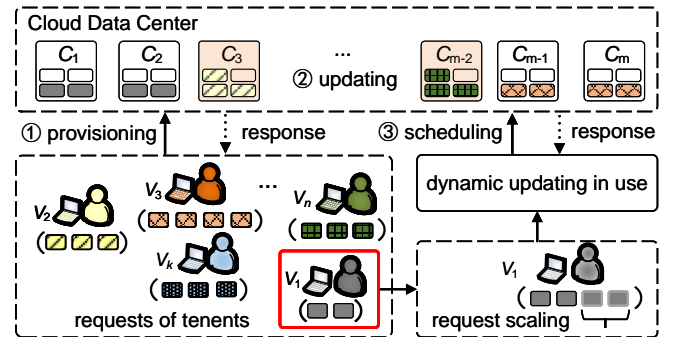

Fig. 1. An illustration of the dynamic request updating in multi-tenant cloud-based DCN.

mize the elasticity while achieving rapid response to multi-tenants in the DCNs.

We give an example that illustrate our work in this paper, some assumptions and notations are not explicitly stated and will be explained in a later section. As shown in Figure. 1, we consider a multi-level tree-based network with m physical machines as the data center architecture. The capacity of each physical machine is slotted, and each slot can only host one virtual machine. We assume that several virtual clusters are running in the operating data center, which are distributed on different servers resulting in variations in the remaining resources. We separate the scenarios of resource requesting of virtual clusters into two categories which are initial provisioning for new newcomers and dynamic updating for those already in use. Here, we desire to find a provisioning strategy for virtual clusters that achieving quick response while also supporting maximum

---

- *Shuaibing Lu and Juan Fang are with the Faculty of Information Technology, Beijing University of Technology, Beijing, China, 100124.*
  E-mail: lushuaibing@bjut.edu.cn.
- *Jie Wu is with the Department of Computer and Information Sciences, Temple University, 1805 N. Broad .Street, Philadelphia, PA 19122.*
  E-mail: jiewu@temple.edu.
- *Haiming Liu is with the School of Software Engineering, Beijing Jiaotong University, Beijing, China, 100091.*
  E-mail: liuhaiming@bjtu.edu.cn.

elasticity without resorting to reassignment. This problem is non-trivial due to the following unique challenges: (i) It is nontrivial to implement fast provisioning and offer a quick response to numerous customers when several newly arrived virtual clusters arrive simultaneously (# of n). (ii) The initial position is vital for the flexible growth of subsequent resources for virtual clusters. It is nontrivial to improve their elasticity under the high dimensions caused by the concurrent requests from enormous amount of virtual clusters and the huge volume of the cloud-based data center. (iii) Additionally, due to fluctuations in demand, tenants who have already provisioned resources in the data center may request resource scaling during the operation. For instance, in Figure. 1, tenant V_1 initially requests 2 VMs, but when the demands varies over the span of the operating period, the number scales to 4. It is challenging to deal with the dynamic scaling of requests that can realize adaptability.

In this paper, we introduce a novel dynamic updating framework with elastic resource scheduling. Instead of using heuristic algorithms or simple deep reinforcement learning as the existing work does, we deal with the resource provisioning problem of multi-tenant and realize the dynamic adjustment for the elastic updating requests at the same time, which consists of two stages. In the first stage, we try to find a feasible provisioning scheme for the multi-tenant with a rapid response by relaxing partial constraints. In the second stage, we attempt to updating the virtual clusters of multi-tenant that have been provisioned by using deep reinforcement learning, so as to improve the elasticity. However, due to the large scale of the scenario, we need to design an efficient solution that improving the training speed of the neural network in high-dimensional space. Our contributions can be summarized as follows:

- We investigate the virtual clusters provisioning problem in multi-tenant cloud-based DCNs with hose model, and we propose to maximize the elasticity by considering the limitation on computation and communication resources.
- We make a theoretical and experimental study of the commonly used methods that are appropriate for provisioning of virtual clusters, and we analyze the insights that produce high complexity and slow convergence.
- We introduce a novel dynamic updating framework with elastic scheduling that make it possible for multi-tenant cloud-based DCNs to provide scalable resources in two stages. We construct a heuristic rapid provisioning scheme in the first stage to realize the real-time response to multi-tenant virtual clusters, and we prove the optimality under the single computation resource constraint.
- Based on that, we present an online dynamic updating method based on deep reinforcement learning to enhance the adaptability of virtual clusters that are running or scaled during the second stage. In order to avoid the high dimensions caused by the large scales of tenants and the DCNs, we train a fully connected neural network by creating a new feasible action set to realize the reduction and it approximates the policy based on a proposed aggressive objective

selection method to improve training speed.
- We conduct various evaluations with several state-of-the-art algorithms under different topologies on the basic setting that refers to the observations. The results are shown from different perspectives to provide conclusions.

The remainder of this paper is organized as follows. Section 2 surveys related works. Section 3 describes the model, problem formulation, and motivation. Section 4 investigates the problem by proposing an efficient framework. Section 5 presents the evaluations. Section 6 concludes the paper.

2 RELATED WORK

Recently, the main research point of virtual cluster provisioning in DCNs includes reliability, energy consumption, traffic changing, and congestion control. However, with the explosion of increasing types and scales of the requests by users, the problem of elasticity in the data center has also been focused. The solutions are mainly divided into three categories introduced in the following.

2.1 Elastic Scheduling by Extending Resources

With the explosion of increasing types and scales of requests by users, the problem of elasticity in the DCN has been focused. There have been a few recent work on elastic resource provisioning by extending physical resources. Rui et al. [7] and Naskos et al. [9] showed a probability model and a cost-aware method to analyze the bottleneck in multi-layer cloud applications, and they proposed a method to meet the elastic scaling of the data center. Farokhis et al. [10] designed a two-layer traffic-aware transmission algorithm, which can effectively solve the problem of virtual machine placement and ensure the large-scale elastic scaling of potential user resources. Lin et al. [11] provided a unified framework that integrates the representation of the logic graphs to maintain regular and reliable operation of data center networks and transmit data between servers. Fan et al. [12] presented an adaptive path-finding algorithm for establishing virtual links between any two nodes in the data center network. Wang et al. [13] realized the elastic scaling of cloud-based DCN by adjusting the size of CPU and memory. Chowdhury et al. [14] analyzed the problem of elastic management based on virtual cluster service, separated resource allocation from service management, and provided the ability of elastic service to adapt to dynamic workload changes. The above works realized the elastic resource provisioning by adding new instances (VMs, containers or application instance modules, etc.) or adjusting the size of the instance in itself during the runtime, which can solve the problem of the insufficient physical resources caused by the dynamic scaling. However, it is difficult to avoid the low utilization and high cost caused by the uncertainty of requesting types and scales.

2.2 Elastic Scheduling by Designing Heuristic Strategies

Quite a few works have been carried out on the elastic resource provisioning problem by designing heuristic strategies. Alfonso et al. [15] proposed an open-source virtual

TABLE 1
Symbols and Definitions.

Symbols	Definitions
$\mathbf{G}$	Topology of DCN, where $\mathbf{G} = \{\mathbf{C}, \mathbf{L}\}$.
$\mathbf{C}$	Set of physical machines, where $\mathbf{C} = \{C_i\}$.
$\mathbf{L}$	Set of physical links, where $\mathbf{L} = \{L_{ij}\}$.
$c_i, \hat{c}_i$	Total and rest available resource of C_i.
$l_{ij}, \hat{l}_{ij}$	Total and rest available resource of L_{ij}.
$\mathbf{C}(L_{ij})$	Set of physical machines under the link L_{ij}.
$\mathbf{V}$	Set of virtual clusters, where $\mathbf{V} = \{V_k\}$.
$v_{k(h)}$	The hth VM of virtual cluster V_k.
$\tau_{v_{k(h)}}^{C_i}$	A boolean variable that indicates $v_{k(h)}$ placed on C_i.
E, E_m, E_l	Elasticity of $\mathbf{G}$, physical machines, and physical links.
δ_i	Estimated divided factor of C_i.
s, a	State and action space.
$\hat{\mathbf{A}}$	Feasible action set.

cluster framework based on the DCN, which analyzed the dynamic changes of virtual clusters in the running process to minimize the cluster consumption and meet the computing demands of users. Kholidy et al. [16] and Guerrero et al. [17] developed a prediction method based on swarm intelligence, which realized the optimal allocation of multi-dimensional resources requested by tenants using particle swarm optimization algorithm and genetic algorithm. Qaddoum et al. [18] proposed an elastic resource scheduling strategy based on load prediction, and solved the problem of computing load fluctuation in big data streaming computing platforms by designing a resource allocation method based on a modified adaptive neural network. Wang et al. [19] presented an elastic resource provisioning scheme to help service providers pay less cost with users' QoS guarantee in clouds. Gao et al. [30] put forward a congestion-aware scheduling scheme to determine the priority of flows based on the latest network congestion in data center networks. Tran et al. [31] designed heuristic traffic-aware virtual network function placement and migration algorithms to minimize the total network traffic in policy-preserving data centers. Fei et al. [32] present a method of elastic resource provisioning using data clustering in cloud service platform. Mina et al. [33] proposed a non dominated set construction algorithm based on task fluctuation to realize the multi-objective dynamic scheduling strategy of elastic cloud resources. Most of the above researches adopt centralized deployment, which prefers to use heuristic or meta-heuristic algorithms to construct resource allocation architecture and realizes elastic allocation of cloud data center resources by software architecture adjustment.

2.3 Elastic Scheduling based on Reinforcement Learning

Reinforcement learning (RL), as one of the paradigms and methodologies of machine learning, uses agent learning strategies in the interaction of the environment to achieve specific goals [20], [29], [34], [35]. Deep reinforcement learning combines the perception ability of deep learning with the decision-making ability of reinforcement learning. In recent years, deep reinforcement learning has been successfully applied in the game, robot control, simulation, optimization, and scheduling [21]. There are many applications of the resource provisioning problem in the cloud data centers. Bitsakos et al. [22] used the deep Q-network algorithm

of deep reinforcement learning to optimize the allocation of elastic resources in DCN with current multidimensional states of virtual clusters as the input. Liang et al. [23] proposed a method based on advantage actor critical deep reinforcement learning, which effectively updates parameters by designing an adaptive scheduling algorithm to realize the resource provisioning in the cloud-based DCNs. Liu et al. [24] developed a reinforcement learning-based framework that adopts neural networks with an overall consideration of data movement and analytical latency, and they trained with a variant of q-learning to solve the data placement problem. Nouris et al. [25] proposed and implemented a controller based on RL, which can not only realize the rapid expansion of resources in the cloud data center, but also save costs by shutting down redundant servers. Chen et al. [26] developed a two-set deep reinforcement learning system to solve the problem of traffic elastic expansion in the cloud data center. By collecting flow information on the terminal host, the data center traffic can be controlled online. Ying et al. [27] proposed a scheduling program based on deep reinforcement learning, which used a cross-entropy method to train a fully connected neural network to realize VM migration before resource updating. Due to the complexities of applications and the scaling of DCNs may generate huge state and action spaces, applying deep reinforcement learning methods directly on the virtual cluster provisioning for multi-tenant will result in a slow convergence rate or even be unable to converge.

3 MODEL AND PROBLEM FORMULATION

In this paper, we study the dynamic virtual cluster updating problem in multi-tenant cloud-based data center network through online scheduling, which attempts to realize rapid respond and high elasticity. In this section, we start with the descriptions of data center model and the virtual cluster model. The problem is also formulated.

3.1 Data Center Network Model

In this paper, the substrate topology of the data center network is defined as tree-structured, i.e. $\mathbf{G} = \{\mathbf{C}, \mathbf{L}\}$. Let $\mathbf{C}$ denote the set of physical machines, where $\mathbf{C} = \{C_i\}$. We use $|\mathbf{C}|$ to represent the total number of physical machines in $\mathbf{G}$. Here, we choose C_i to denote the ith machine. The capacity of C_i is c_i, and the remaining resources is $\hat{c}_i$. Based on our data center topological structure, we use $\mathbf{L}$ to denote the set of physical links, where $\mathbf{L} = \{L_{ij}\}$. Here, L_{ij} denotes the jth link on the ith level, and the capacity of L_{ij} is l_{ij} in $\mathbf{G}$. For each link being used, we use $\hat{l}_{ij}$ to represent the remaining available communication resources.

3.2 Virtual Clusters

We use virtual clusters to denote the demands of multi-tenant, which is denoted by $\mathbf{V} = \{V_i\}$. Each virtual cluster represents the requirements on both computing and communications resource which consists of a set of virtual machines (VMs) and one virtual switch [3], where $V_k = \{v_{k(h)}\}$. Let $v_{k(h)}$ denote the hth VM of virtual cluster V_k. We use $|V_k|$ to denote the total number of VMs in V_k, and we suppose that each VM has B_kGbps unit bandwidth
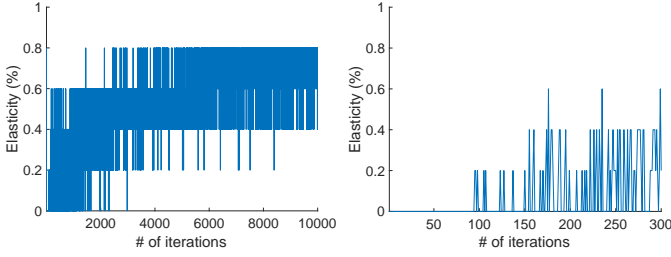
(a) The elasticity within 10^4 iterations.

(b) The fluctuation from 0 to 300 iterations.

Fig. 2. Illustration of a slow convergence condition based on a straight-forward deep reinforcement learning method.

demand between VMs and the virtual switch of virtual cluster V_k. The total number of VMs in set $\mathbf{V}$ is denoted as $|\mathbf{V}|$, where $|\mathbf{V}| = \sum_{k=1}^{n} |V_k|$. In this paper, we consider the virtual clusters with hose model which means that each customer specifies a set of endpoints to be connected with a common endpoint-to-endpoint performance guarantee [5]. Virtual clusters are independent from one another and only interact with the inside VMs, there is no interaction between external virtual clusters. Here, we use communication demand to indicate the maximum bandwidth resources that each VM may simultaneously communicate with the other virtual machines in its own cluster. The communication demand of these components will be 0Gbps since VMs that are housed on the same physical machine do not use link bandwidth resources when communicate with one another. We use $f(V_k, C_i)$ to indicate the communication demand of VMs provisioning on machine C_i for virtual cluster V_k which deployed across different machines. For each virtual cluster, we use a boolean variable $\tau_{v_{k(h)}}^{C_i}$ to indicate that whether $v_{k(h)}$ placed on C_i. $\sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}$ denotes the number of VMs belonging to C_i. Therefore, we have that the total communication demand of V_k on C_i is $f(V_k, C_i) = \min\{\sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}, |V_k| - \sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}\} \cdot B_k$. Taking V_3 as an example shown in Figure. 1, which is made up of four VMs, each VM in V_3 is able to communicate with the other three. We supposed that each VM has a $B_3 = 1$Gbps unit bandwidth demand of V_3, then $f(V_3, C_{m-1}) = \min\{2, (4-2)\} = 2 \cdot B_3 = 2$Gbps is the result.

3.3 Problem Formulation

This paper focuses on the virtual cluster provisioning problem in multi-tenant DCNs. We use E to denote the combinational elasticity of the DCN, which is expressed as a percentage and used to represent the usage of the minimum remaining resources, which is shown in equation (1). We aim at minimizing the combinational elasticity by considering both computational and communication resources. Here we use E_m to represent the elasticity of the physical machines, which results from calculating the minimum scalable capacity of the physical machines. Similarly, we use E_l to represent the elasticity of physical links which is shown in equation (2). In equation (3), we use λ and φ to represent coefficients reflecting different computational and communication resource requirements. The constraints during the virtual cluster provisioning are shown in equations (4) and (5), which means that the total consumption of computation and communication resources cannot exceed the capacities

of physical machines and links. Here, equation (5) is the communication demands of virtual clusters on link l_{ij}, where $\sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}$ denotes the number of VMs belonging to C_i. Equation 6 is the communication demand of the virtual cluster V_k on C_i. We use the notion of elasticity to measure the potential growth of multi-tenant in terms of both computing and communication resources [5], [6]. The problem formulation is shown as follows:

$$\text{maximize} \quad E = \min\{\lambda E_m, \varphi E_l\} \tag{1}$$

$$\text{s.t.} \quad E_m = \min_i\{1 - \frac{\hat{c}_i}{c_i}\}, E_l = \min_{i,j}\{1 - \frac{\hat{l}_{ij}}{l_{ij}}\} \tag{2}$$

$$0 \le \lambda \le 1 \text{ and } 0 \le \varphi \le 1 \tag{3}$$

$$\hat{c}_i \le c_i \text{ and } \hat{l}_{ij} \le l_{ij} \tag{4}$$

$$\hat{l}_{ij} = \sum_{C_i \in \mathbf{C}(L_{ij})} \sum_{V_k \in C_i} f(V_k, C_i) \tag{5}$$

$$f(V_k, C_i) = \min\{\sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}, |V_k| - \sum_{h \in V_k} \tau_{v_{k(h)}}^{C_i}\} \cdot B_k \tag{6}$$

3.4 Motivation

3.4.1 High time complexity under DP

Based on the problem formulation in Equations (1) to (4), the numbers of variables and constraints are large which means that the virtual cluster provisioning problem in multi-tenant DCNs cannot be efficiently solved by the simplex or eclipse methods. A dynamic programming (DP) scheme has been proposed in [3], and the time complexity is proved to be $O(2^h \cdot \prod_{j=1}^{|\mathbf{V}|}(|V_j|+1)^{h-j})$, where h is the height of the DCN. We take the data center of Alibaba [4] as an example, which is constructed by 4034 physical machines. We suppose that there are only two tenants with two virtual clusters, and each one only contains a single VM. According to the calculation method, the time complexity under DP will be higher than 2^{4036}, which is huge.

3.4.2 Slow convergence under high dimension

To avoid the high complexity brought by the DP method, we consider to use deep reinforcement learning, such as deep Q-learning (DQN), to solve the virtual cluster provisioning problem in multi-tenant DCNs. We suppose the state space $s_t = [\hat{c}_1, \hat{c}_2, ..., \hat{c}_i, ..., \hat{c}_m, |\hat{V}_1|, |\hat{V}_2|, ..., |\hat{V}_j|, ..., |\hat{V}_n|]$, where $\hat{c}_i$ denotes the remaining resource of ith physical machine and $|\hat{V}_j|$ denotes the number of remaining VMs of jth virtual cluster. The action space is $a_t = [s_{1(1)|}, ..., s_{1(m)}, s_{2(1)}, ..., s_{2(m)}, ..., s_{n(1)}, ..., s_{n(m)}]$, where $s_{i(j)}$ is a boolean variable that denotes whether the VM provisioning on the jth physical machine. Based on that, the maximum times of calculation is $C(\sum_{j=1}^{n} |\hat{V}_j|, \sum_{i=1}^{m} \hat{c}_i)$. The reward mechanism is defined in Algorithm 2, which is $r = (E(S_{t+1}) - \bar{E})/(\bar{E} + \xi)$. We take 6-layer topology as an example as shown in the sub-figure (a) of Figure. 2, and the capacity of the physical machine is defined as $c_i = 8$. We suppose that there are only two virtual clusters V_1 and V_2, where the number of VMs are $|V_1| = 5$ and $|V_2| = 5$, respectively. The result is shown in Figure. 2(a). We can see that the elasticity of the DCN is close to convergence when the number of iterations reaches 10^4. In addition, the range of the elasticity fluctuates greatly in the iterations
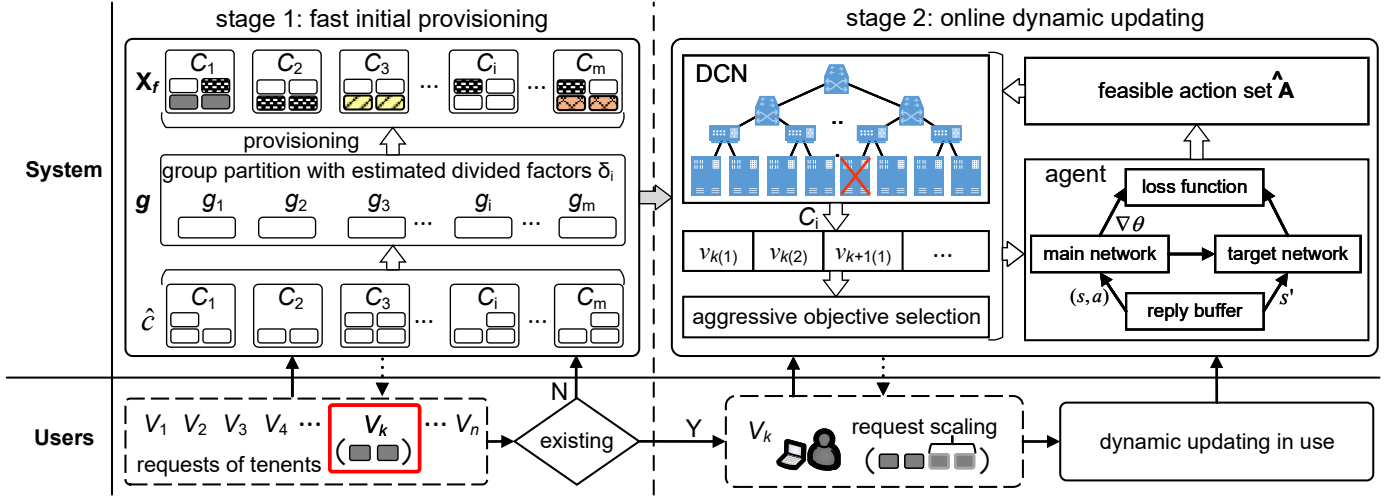
Fig. 3. The overview of DUES

from 6×10^3 to 10^4. We enlarge the value of elasticity within 300 iterations in Figure. 2(b), which has no tendency of convergence and fluctuates sharply. Thus, we can see that the convergence time will be extremely slow if the deep reinforcement learning method is used to directly search for and learn the best solution of the virtual cluster provisioning problem for multi-tenant.

4 DYNAMIC UPDATING FRAMEWORK WITH ELASTIC SCHEDULING

In this section, we show the detail of our novel online dynamic updating framework with elastic scheduling (DUES) which constructed by two stages to realize the rapid response and high elasticity.

4.1 Overview

The main idea of DUES is to realize real-time response to multi-tenant requests for provisioning and upgrading while maximizing the elasticity of the cloud-based DCN. The overview of DUES, which comprises of two stages, is shown in Figure 3. In the first stage, we propose a heuristic scheme to realize the fast provisioning for multi-tenant and analyze its optimality and complexity. We take the arriving requests of multi-tenant virtual clusters as the input, and the output is the initial provisioning scheme which also converts to the input of the second stage. In the second stage, we propose a online dynamic updating strategy based on deep reinforcement learning to improve the combinational elasticity of the cloud-based DCN. Since the main drawback of the simple deep reinforcement learning method is that it will result in huge state and action spaces when applied to the virtual cluster provisioning problem, we introduce a new definition which is the feasible action set. Based on that, we train a fully connected neural network to realize the reduction and it approximates the policy based on a proposed aggressive objective selection method to improve training speed. The detailed description are shown as follows.

4.2 Stage 1: Fast Initial Provisioning

In this subsection, we introduce a multi-tenant fast initial provisioning scheme (MFIP) for virtual clusters which is

shown in Algorithm 1. The insight of our scheme is to identify the partition for virtual clusters based on the computing resource of the DCN. The input in Algorithm 1 is the set of virtual clusters $\mathbf{V}$, and the output is the fast provisioning scheme $\mathbf{X}_f$. In lines 1 and 2, we first check the feasibility of virtual clusters by comparing $|\mathbf{V}|$ with $\hat{c}$. Here, we use $\hat{c}$ to represent the total remaining resources where $\hat{c} = \sum_{i=1}^{m} \hat{c}_i$. If the remaining computing resources can accommodate the virtual clusters of $\mathbf{V}$, where $|\mathbf{V}| \leq \hat{c}$, Algorithm 1 continues. Otherwise, the requests of set $\mathbf{V}$ will be rejected. In lines 3 to 5, we start to calculate the estimated number of accommodation based on the computing capacities of each server in $\mathbf{G}$. Here, we introduce a new definition of the estimated divided factor.

Definition 1 (Estimated divided factor). Let δ_i denote the estimated divided factor of C_i and $\delta_i = \hat{c}_i / \sum_{i=1}^{m} \hat{c}_i$, where $\hat{c}_i$ denotes the rest available physical resources.

In line 3, we first initialize the group partition with estimated divided factors. Then we calculate the capacity of each group, which is the maximum amount of provisioning VMs on server i, i.e, $g_i = \delta_i \cdot |\mathbf{V}|$. The value of g_i is an integer that rounds down with $g_i = \lfloor \delta_i \cdot |\mathbf{V}| \rfloor$ to avoid overflowing, which involves reducing to the nearest integer even if the fractional part is larger than or equal to 0.5. We suppose that if an upward value or rounding method is used to obtain the value of g_i, it is possible that the total available resources on the servers of the groups will be higher than the total number of requests, resulting in an overflow error where the free position information exceeds the total number of requests. Here, we highlight the potential impact through using a straightforward example. We suppose that there are only 3 available servers left in the data center, and each has $\hat{c}_1 = 2$, $\hat{c}_2 = 10$, and $\hat{c}_3 = 8$ remaining resources, respectively. We assume that the total amount of requests is $|\mathbf{V}| = 15$ altogether of virtual clusters at this time, we will obtain $g_1 = 2$, $g_2 = 8$, and $g_3 = 6$, where $\sum_{i=1}^{i=3} g_i = 16 > |\mathbf{V}|$ regardless of whether rounding up or normal rounding is used, which results in overflow. In line 6, we update the number of VMs of multi-tenant virtual clusters $|\mathbf{V}| = |\mathbf{V}| - \sum_{i=1}^{m} g_i$. In lines 7 to 9, we resize some groups for the rest of the queries in $\mathbf{V}$. If $|\mathbf{V}| \neq 0$, which

means there are remaining VMs that cannot be covered, we choose the physical machine with the maximum available resources by $\arg\max_{C_i \in C} \hat{c}_i$ and increase the estimated number of accommodations by $g_i = g_i + 1$ in lines 8 and 9. Then, we update the total number of requests in line 10. In line 11, we update groups in set $\mathbf{g}$ with descending order based on their sizes, where $\mathbf{g} := descending(\mathbf{g})$. Then we start the provisioning process, which chooses request V_j with maximum $\arg\max_{V_j \in \mathbf{V}} N_j$ and matches V_j into group g_i. If the number of VMs $|V_j|$ is over the estimated number of accommodation in g_i, we place a part of VMs according to the size of g_i. Then we update $g_i = 0$ and remove g_i from set $\mathbf{g}$. Otherwise, we update set $\mathbf{g}$ with $g_i = g_i - |V_j|$. After that, we update $\mathbf{g} := descending(\mathbf{g})$ in line 19. Line 20 returns the provisioning scheme $\mathbf{X}_f$. The time complexity of Algorithm 1 is $O(m^2 \cdot |\mathbf{V}|)$.

Theorem 1. The total communication demand of virtual clusters $\mathbf{V}$ with MFIP is minimum in $\mathbf{G}$ under the single constraint $\hat{c}_i \leq c_i$.

Proof: There are two steps in MFIP, which are initializing the estimated groups of $\mathbf{G}$ and the identifying partition for multiple virtual clusters. In the first step, the groups are estimated by considering converting multiple virtual clusters to a single one. The partition of each group is found by calculating the capacities based on the physical machines in $\mathbf{G}$ that is optimal, which has been proved in [28]. Thus, we only need to prove that the identifying partition for $\mathbf{V}$ obtained the minimum total communication demand. We suppose that the set of estimated groups $\mathbf{g} = \{g_i\}$ has been updated with descending order. If the demands of all virtual clusters in $\mathbf{V}$ with the same order are lower than groups in $\mathbf{g}$, the total communication demand of virtual clusters $\mathbf{V}$ will be 0 which is minimum. If there is existing $V_i \subseteq \mathbf{V}$ larger than the size of group g_i, where $|V_i| > |g_i|$, $V_i \subseteq \mathbf{V}$ will be divided into several parts. We discuss the case as follows.

We suppose that the fast provisioning $\mathbf{X}_f$ is $(V_i \to g_i)_{|V_i|>|g_i|}$, the total communication demand is $\min\{|V_i| - |g_i|, |g_i|\}$. Assuming that the provisioning with minimum communication demand (MCD) of V_i is group g_j, then we have $(V_i \to g_j)_{|V_j|>|g_j|}$. The total communication demand will be $\min\{|V_i| - |g_j|, |g_j|\}$. Here, we prove by contradiction which assumes that $|g_i| > |g_j|$, then we have four possible scenarios. (i). $\min\{|V_i| - |g_i|, |g_i|\} = |V_i| - |g_i|$ under MFIP and $\min\{|V_i| - |g_j|, |g_j|\} = |V_i| - |g_j|$ under MCD. Since we suppose MCD has the minimum communication demand, then we have that $|V_i| - |g_j| < |V_i| - |g_i|$, i.e. $|g_i| > |g_j|$, which contradicts with the assumption $|g_j| < |g_i|$. (ii). $\min\{|V_i| - |g_i|, |g_i|\} = |V_i| - |g_i|$ under MFIP and $\min\{|V_i| - |g_j|, |g_j|\} = |g_j|$ under MCD. Suppose the group that provisioning the $V_i \setminus g_j$ request is g_k, where the rest capacity $|V_i| - |g_i| < |\hat{g}_k| < |g_i|$. The total communication demand will be $2(|V_i| - |g_i|)$. Since we suppose MCD has the minimum communication demand where $|g_j| < |V_i| - |g_i|$, then we have that $|V_i| - |g_j| > g_i$. Since $|\hat{g}_k| < |g_i|$, we have $|\hat{g}_k| < |g_i| < |V_i| - |g_j|$. The MCD will be at least $2|g_j| + 2|g_k|$, however, $|V_i| - |g_i| < |\hat{g}_k|$, then we have $2|g_j| + 2|g_k| > 2|g_j| + 2(|V_i| - |g_i|)$, which contradict with the assumption. For the remaining two cases (iii) ($\min\{|V_i| - |g_i|, |g_i|\} = |g_i|$ under MFIP and $\min\{|V_i| - |g_j|, |g_j|\} = |g_j|$) and (iv) ($\min\{|V_i| - |g_i|, |g_i|\} =$

$|g_i|$ under MFIP and $\min\{|V_i| - |g_j|, |g_j|\} = |V_i| - |g_j|$), the proof process is the same as (a) and (b). In summary, the total communication demand of virtual clusters $\mathbf{V}$ with MFIP is minimum under the single constraint $\hat{c}_i \leq c_i$. ∎

Algorithm 1 Multi-tenant Fast Initial Provisioning Scheme (MFIP)

Input: Set of multi-tenant requests $\mathbf{V}$;
Output: Fast initial provisioning scheme $\mathbf{X}_f$;
1: **if** $|\mathbf{V}| < \hat{c}$ **then**
2:    **return** False;
3: **for** each server in $\mathbf{G}$ **do**
4:    Initialize group partition with the estimated divided factor δ_i;
5:    Calculate the capacity of each group g_i;
6: Update the virtual clusters $|\mathbf{V}| := |\mathbf{V}| - \sum_{i=1}^{m} g_i$;
7: **while** $|\mathbf{V}| \neq 0$ **do**
8:    Choose the physical machine with $\arg\max_{C_i \in C} \hat{c}_i$;
9:    Update $g_i := g_i + 1$;
10:   Update $|\mathbf{V}| := |\mathbf{V}| - 1$;
11: Update $\mathbf{g} := descending(\mathbf{g})$;
12: **for** each group g_i in $\mathbf{G}$ **do**
13:   Choose request V_j with $\arg\max_{V_j \in \mathbf{V}} |V_j|$;
14:   Matching V_j into group g_i;
15:   **if** $|V_j| > g_i$ **then**
16:      $|V_j| := |V_j| - g_i$;
17:      Update $g_i := 0$ and remove g_i from set $\mathbf{g}$;
18:   **else**
19:      Update set $\mathbf{g}$ with $g_i := g_i - |V_j|$;
20:   Update $\mathbf{g} := descending(\mathbf{g})$;
21: **return** Initial Provisioning $\mathbf{X}_f$;

4.3 Stage 2: Online Dynamic Updating Strategy

In this subsection, we propose an online dynamic updating strategy based on deep reinforcement learning. The main idea is to realize the dynamic updating requests from multi-tenants inside each equal time slot after dividing continuous time into equal slices. This process intends to identify the bottleneck of the DCN for each time slot in the context of the fast initial provisioning $\mathbf{X}_f$ and choose VMs to readjust based on an aggressive upgrading strategy for the objective selection. Here, it is worth noting that when a request is produced at a certain time slot, it is necessary to verify beforehand whether the tenant is already located in the data center. If the tenant is already in the data center, it goes directly to the second stage of dynamic online updating; if not, it must go through the first stage of fast initial provisioning. We define the bottleneck as follows.

Definition 2 (bottleneck). The bottleneck b^* is a vector representing the location of physical machine or link with the minimum elasticity of $\mathbf{G}$.

The core component of the agent is to design a policy where it provides the probability distribution over the action space a and the state space s.

4.3.1 Deep Reinforcement Learning Formulation

In order to describe the environment of the DCN concisely and correctly for the agent, the state space should include

---

Algorithm 2 Reward Updating Mechanism

Input: Variable num;
Output: Reward $r(s_t, a_t)$ and termination variable Γ;
 1: **if** $E(s_{t+1}) \leq E(s_t)$ **then**
 2:   $num := num + 1$;
 3: **if** $E(s_{t+1}) > E(s_t)$ **then**
 4:   $r := (E(s_{t+1}) - \bar{E})/(\bar{E} + \xi)$;
 5: **else if** $E(s_{t+1}) = E(s_t)$ **then**
 6:   $r := E(s_{t+1}) - E(s_t)$;
 7: **else**
 8:   $r := -1$;
 9: **if** $num \geq \Psi$ **then**
10:   $\Gamma := ture$;
11: **else**
12:   $\Gamma := false$;
13: **return** Reward $r(s_t, a_t)$ and termination variable Γ;

---

the knowledge of the usage on the physical resources, the status of requests from multiple tenants, and the information of the updating virtual cluster. So the state is designed as follows.

Definition 3 (state). The state s_t is a vector consisting of $s_t = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_m]_t$, where $\mathbf{h}_i$ denotes the provisioning list on server c_i at time slot t. The provisioning list $\mathbf{h}_i$ records the number of virtual clusters placed on this server, where $\mathbf{h}_i = [v_1, v_2, v_i, ..., v_k]$.

We consider realizing the dynamic updating by training the agent which needs to choose a destination physical machine for each adjusting VM. The action a_t is designed as follows.

Definition 4 (action). The action space $a_t = [\mathbb{C}_1, \mathbb{C}_2, ..., \mathbb{C}_m]_t$ is the updating action, where $\mathbb{C}_i = 0$ or $\mathbb{C}_i = 1$ means that the target location of adjustment is on C_i or not at time slot t.

The objective of the agent is to find a provisioning scheme for multi-tenants that maximizes the elasticity of the DCN. For each episode, we decide the provisioning of VMs for the tenants by choosing an action. After that, the agent will get a reward $r(s_t, a_t)$ at time slot t with state s_t after executing action a_t. In our problem, the value of this reward cannot determine the final elasticity until all requests of tenants are provisioned. The reason is that the virtual cluster only communicate with VMs on their own, which means that although we make the adjustment decision for VMs one by one, the final elasticity is determined until all virtual clusters finish provisioning. Here is the specific definition.

Definition 5 (Reward). The reward r is decided by the value of elasticity which defined in three cases, where

$$r := \begin{cases} (E(s_{t+1}) - \bar{E})/(\bar{E} + \xi) & E(s_{t+1}) > E(s_t) \\ E(s_{t+1}) - E(s_t) & E(s_{t+1}) = E(s_t) \\ -1 & E(s_{t+1}) < E(s_t) \end{cases} \quad (7)$$

The updating mechanism of the reward shown in Algorithm 2, we initialize a variable num to record the times that the elasticity decreases after adjusting the VMs in one episode. The output of this function is the reward after choosing action a_t and the value of the termination variable Γ. In line 1, we first compare the value of elasticity under the

---

Algorithm 3 Feasible Actions Set Construction Function

Input: Fast initial provisioning $\mathbf{X}_f$;
Output: Feasible actions set $\hat{\mathbf{A}}$;
 1: Calculate E under the fast initial provisioning $\mathbf{X}_f$;
 2: Find the location of the bottleneck with minimum elasticity;
 3: **if** $E = E_m^{C_i}$ **then**
 4:   Remove the bottleneck C_i from set $\hat{\mathbf{A}} = \{\mathbf{A}/C_i\}$;
 5: **else**
 6:   Remove the physical machines in set $\mathbf{C}(L_{ij})$ under the bottleneck link L_{ij} from set $\hat{\mathbf{A}} = \{\mathbf{A}/\mathbf{C}(L_{ij})\}$;
 7: **return** Feasible Actions Set $\hat{\mathbf{A}}$;

---

state s_t and s_{t+1}. If $E(s_{t+1}) \leq E(s_t)$, it means that the elasticity after choosing action a_t will decrease, we will record it using $num := num + 1$. After that, we start to define the reward updating mechanism under different cases. In lines 3 to 4, when there is an increment that $E(s_{t+1}) > E(s_t)$, the reward will be defined as $r := (E(s_{t+1}) - \bar{E})/(\bar{E} + \xi)$, where $\bar{E}$ is the baseline elasticity after the fast provisioning $\mathbf{X}_f$. ξ is a factor that avoid the denominator obtaining zero, where $0 < \xi \leq 1$. In lines 5 to 6, if the $E(s_{t+1}) = E(s_t)$, the reward will be defined as $r := 0$. Otherwise, the reward will be defined as $r := -1$ under the $E(s_{t+1}) > E(s_t)$ case in lines 7 to 8. Since the reward of one action cannot determine the final result, the reward value $r := -1$ cannot represent that the total provisioning order which is bad. However, if the bad cases continue to happen, which means that the agent always chooses the action with $r := -1$, this episode will be terminated when $num \geq \Psi$. Here, we use Ψ to denote a threshold that is determined by the structure of the DCN, which is less than or equal to the number of physical machines, i.e., $\Psi \leq |C|$. Line 13 returns the reward $r(s_t, a_t)$ and the termination variable Γ.

4.3.2 Construction of Feasible Actions Set

In order to reduce the high dimensions caused by the large scales of tenants and the DCNs, we introduce a definition of the feasible action set.

Definition 6 (Feasible Actions Set). Let $\hat{\mathbf{A}}$ indicate the feasible action set of $\mathbf{V}$, which only include the target VMs that have positive effect on optimize the combinational elasticity of $\mathbf{G}$.

Based on that, we further propose a method to construct a feasible action set $\hat{\mathbf{A}}$. The main idea is to remove the invalid optional targets which exist under the bottleneck to improve the learning speed of the agent. The detail description is shown in Algorithm 3. In line 1, we first calculate the elasticity E under the fast initial provisioning $\mathbf{X}_f$, and we find the location of bottleneck. If the bottleneck is located on the physical machine $E = E_m^{C_i}$, where $E_m^{C_i}$ is the elasticity of C_i, i.e., $E_m^{C_i} = 1 - \frac{\hat{c}_i}{c_i}$. Then, we remove the optional target C_i from set $\mathbf{A}$, where $\hat{\mathbf{A}} = \{\mathbf{A}/C_i\}$. If the bottleneck is located on the physical link $E = E_l^{L^{ij}}$, where $E_l^{L^{ij}}$ is the elasticity of L_{ij}, i.e., $E_l^{L^{ij}} = 1 - \frac{\hat{l}_{ij}}{l_{ij}}$. Then, we remove set $\mathbb{C}_{L_{ij}}$ under the bottleneck link L_{ij} from set $\hat{\mathbf{A}} = \{\mathbf{A}/\mathbf{C}(L_{ij})\}$. Here, we use $\mathbf{C}(L_{ij})$ to denote the set of physical machines under the physical link L_{ij}. Line 7 returns the feasible action set $\hat{\mathbf{A}}$.

Algorithm 4 Aggressive Objective Selection Algorithm

Input: State s under the fast initial provisioning $\mathbf{X}_f$;
Output: VM that needs to be adjusted $v_{k(h)}$;
 1: Same as Algorithm 3 in lines 1 to 2;
 2: **if** $E = E_m^{C_i}$ **then**
 3:   Select VM v_k with the maximum communication demand on C_i to adjust;
 4: **else**
 5:   Choose the physical machine C_w in set $\mathbf{C}(L_{ij})$ under bottleneck link $E_l^{L_{ij}}$ with minimum elasticity $E_m^{C_w}$;
 6:   Select VM $v_{k(h)}$ with the maximum communication demand on C_w to adjust;
 7: **return** VM that needs to be adjusted $v_{k(h)}$;

4.3.3 Aggressive Objective Selection

In a given episode, the agent choose an action from the set $\hat{\mathbf{A}}$ that is detailed in Algorithm 3. This action can only determine the destination that we can adjust the VMs, however, which VM is selected to be adjusted cannot be determined by the action a_t. Here, we design an aggressive adjusted objective selection algorithm to recognize which VM is being adjusted based on its current policy. Then the environment will return a reward r_t to the agent and transit to s_{t+1}. In Algorithm 4, the input is the state s under the fast initial provisioning $\mathbf{X}_f$ and the output is the VM v_k that is selected to be adjusted. In lines 1 to 2, we initialize the elasticity E under the fast provisioning $\mathbf{X}_f$ and find the bottleneck same as Algorithm 3. If the bottleneck is located on C_i where $E = E_m^{C_i}$, we select VM v_k with the maximum communication demand on C_i to adjust in line 3. Otherwise, if the bottleneck is located on L_{ij} where $E = E_l^{L_{ij}}$, we choose the physical machine C_w under bottleneck link L_{ij} with minimum elasticity $E_m^{C_w}$ in line 5. Based on that, we select VM v_k with the maximum communication demand on C_w to adjust. Line 7 returns the VM v_k that needs to be adjusted.

4.4 Dynamic Updating based on Deep Reinforcement Learning (DU-DRL)

The overview of the dynamic updating strategy based on deep reinforcement learning (DU-DRL) is shown in the right part of Figure. 3. Algorithm 5 summarizes the specific steps. The main idea is to use a deep reinforcement learning agent to perform the dynamic adjustment VM of virtual clusters to maximize the elasticity of the DCN in each time slot. Before we conduct dynamic updates, we collect the status of multi-tenant virtual clusters $\mathbf{V}$ to construct the set of update requests $\mathbf{V}'$. We prioritize processing for tenants who requesting resource release $\mathbf{V}^-$. Then, we reinitialize the resources of cloud-based DCN and update $\mathbf{V}' = \{\mathbf{V}' - \mathbf{V}^-\}$ which serves as input of Algorithm 5. We first initialize some preliminary parameters which include setting the replay memory $\mathcal{D}$ to capacity N and episode terminated variable Γ to $false$. Meanwhile, we initialize the action-value function Q with random weight θ and the target action-value function $\hat{Q}$ with weights $\theta^- = \theta$. In lines 2 to 4, we start to train the agent by running a number of κ episodes with our environment. During each episode, we initialize sequence S

Algorithm 5 Dynamic Updating based on Deep Reinforcement Learning (DU-DRL)

Input: Set of updating requests $\mathbf{V}'$;
Output: Provisioning scheme $\mathbf{X}$;
 1: Initialize $\mathcal{D}$ to N, Γ to $false$, Q with random weights θ, and $\hat{Q}$ with weights $\theta^- := \theta$;
 2: **for** episode from 1 to κ **do**
 3:   Initialize sequence s based on $\mathbf{X}_f$ of Algorithm 1;
 4:   Preprocessed sequence $\phi_1 = \phi(s_{\mathbf{X}_f})$;
 5:   **while** $\Gamma = false$ **do**
 6:     Build feasible actions set $\hat{\mathbf{A}}$ based on Algorithm 3;
 7:     With probability ε select a random action $a_t \in \hat{\mathbf{A}}$;
 8:     Otherwise select $a_t = argmax_a Q(\phi(s_t), a; \theta)$;
 9:     Choose the adjusted objective based on Algorithm 4;
10:     Execute action a_t in emulator and update r and Γ based on Algorithm 2;
11:     Set $s_{t+1} = s_t$, a_t, x_{t+1} and preprocess $\phi_{t+1} = \phi(S_{t+1})$.
12:     Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$;
13:     Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$.
14:     **if** episode terminates at step $j + 1$ **then**
15:       Set $y_j = r_j$;
16:     **else**
17:       Set $y_j = r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-)$;
18:       Perform a gradient descent step on $(y - Q(\phi_j, a_j; \theta))^2$ with respect to the parameters θ.
19:       Every C steps reset $\hat{Q} = Q$;
20: **return** Provisioning scheme $\mathbf{X}$;

TABLE 2
Step-by-step calculation for the example of DUES.

$\mathbf{X}_f$	$(0 \rightarrow C_1), (2V_2 \rightarrow C_2), (3V_1 \rightarrow C_3), (4V_3 + V_4 \rightarrow C_4)$
$\mathbf{X}$	$(V_4 \rightarrow C_1), (2V_2 \rightarrow C_2), (3V_1 \rightarrow C_3), (4V_3 \rightarrow C_4)$

based on the fast initial provisioning $\mathbf{X}_f$ and preprocess it with $\phi_1 = \phi(s_{\mathbf{X}_f})$ in lines 3 to 4. The training process starts from lines 5 to 20. The process of the adjustment starts from choosing a physical machine from the built feasible actions set $\hat{\mathbf{A}}$ that is produced by Algorithm 3. In line 7, the agent selects a random action $a_t \in \hat{\mathbf{A}}$ with probability ε, otherwise, it will select $a_t = argmax_a Q(\phi(s_t), a; \theta)$ with the maximum Q value in line 8. Since there is a queue of VMs from different virtual clusters provisioning on the chosen physical machine, the agent needs to choose the adjusted objective based on Algorithm 4 in line 9. At each time step, only one of the VMs in this queue is adjusted. Then the agent executes action a_t in the emulator and updates r and Γ based on Algorithm 2. We set $s_{t+1} = s_t$, a_t, x_{t+1}, and preprocess $\phi_{t+1} = \phi(s_{t+1})$, and we store the transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in the replay memory $\mathcal{D}$ in line 12. After that, we sample a random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$. In lines 14 to 17, the agent will calculate the reward after the termination of the episode. The objective of our problem is to maximize the elasticity of the DCN which is consistent with the cumulative reward received by the agent. In line 18 to 19, the agent performs a gradient descent step on $(y - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ, and resets $\hat{Q} = Q$ every C steps. The results are returned
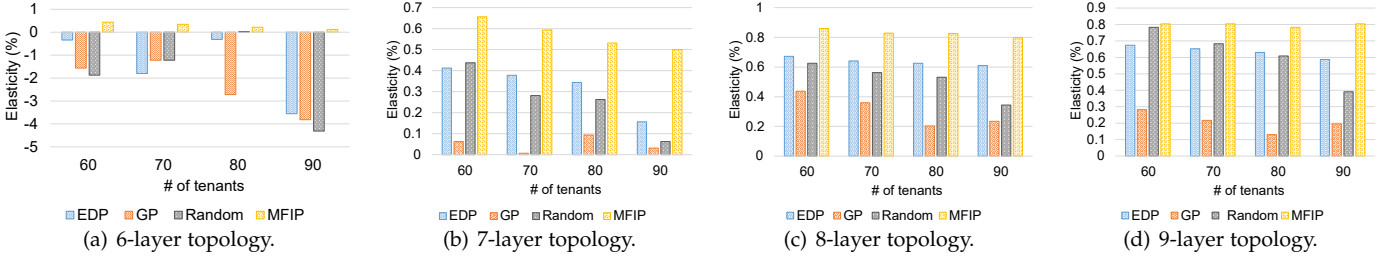
Fig. 4. The elasticities of different DCNs with tenants ranging from 60 to 90 (first-stage).

in line 21. DUES fully considers the tight coupling between virtual machine placement and bandwidth resource allocation for the requests of multi-tenant. When there are resource scaling from tenants running in the data center, our approach can eliminate erroneous optional targets that seem plausible but are prone to elastic bottlenecks by building up the set of feasible actions, which reduces the dimension of the search space and accelerates the learning rate of the agent. In addition, the strategy proposed in our framework enables the agents to accommodate workload fluctuations by recognizing which virtual machine is being adjusted based on its current policy, which results in the rapid allocation of resources under scaling. We use an example to explain the details of DUES. We suppose that the topology of DCN is 3-layer with four physical machines. The remaining computation resource (C_1 to C_4) are $\hat{c}_1 = 2$, $\hat{c}_2 = 5$, $\hat{c}_3 = 7$, and $\hat{c}_4 = 10$. The remaining communication resource of the physical links ($L_{1,1}$ to $L_{2,4}$) are $\hat{l}_{1,1} = 15$, $\hat{l}_{1,2} = 15$, $\hat{l}_{2,1} = 5$, $\hat{l}_{2,2} = 10$, $\hat{l}_{2,3} = 5$, $\hat{l}_{2,4} = 1$. There are four tenants with virtual clusters V_1 to V_4, where $|V_1| = 3$, $|V_2| = 2$, $|V_3| = 4$, $|V_4| = 1$. According to the Algorithm 1, the fast provisioning $\mathbf{X}_f$ is shown in the first row of Table 2, and the elasticity $E_{\mathbf{X}_f} = 0$. The upper link of C_4 is the bottleneck which is highlighted with underline. Then we construct the feasible action set based on Algorithm 3, where $\hat{\mathbf{A}} = C_1, C_2, C_3$. We adjust the VMs based on Algorithm 5, and the provisioning $\mathbf{X}$ is shown in the second row of Table 2.

5 EVALUATIONS

In this section, we conduct extensive simulations and experiments to study the elastic virtual cluster provisioning in multi-tenant DCNs. We develop a prototype of our algorithms using python, which consists of the construction of the data center and the requests of multiple tenants. After presenting the datasets and settings, the results are shown from different perspectives to provide insightful conclusions.

5.1 Basic Setting

We use python to build our prototype on workstation Precision T7910 with Intel Xeon(R) E5-2620 CPU, NVIDIA RTX5000 GPU, $128Gb$ memory, and $2Tb$ hard disk on Linux operating system. We simulate a DCN of k-level binary tree topology, where $k \in \{6, 7, 8, 9\}$. Each physical machine resource is divided into slots, and each slot can only accommodate one virtual machine, which can be easily interpreted to a real configuration. The physical machines have slot

TABLE 3
Hyperparameter Settings.

Hyperparameter	Settings
learning rate α	0.001
e-greedy ϵ	0.98
reward decay γ	0.9
replacing target iterations	200
replay memory $\mathcal{D}$	6000

capacities ranging from 0 to 100, and each of them has a $1Gbps$ link to connect with a higher layer switch. Every switch is connected by a higher layer that has double the communication capacity of the current layer until it reaches the core. The deployments and workload conditions in our basic setting consider the observations from analyzing the public data of Azure [1]. The number of VMs in each virtual cluster ranges in $[10, 20]$, and the core of VM is set to be 1. We consider the number of users ranging from $[60, 90]$. The settings of hyper parameters are listed in Table 3. We test several groups of hyper parameters which include the learning rate $\alpha = \{0.0001, 0.0005, 0.001\}$, and the e-greedy $\epsilon = \{0.95, 0.96, 0.97, 0.98\}$ and so as to others. We choose the group of hyper parameters listed in Table 3 as the experimental setting. In addition to the proposed scheduling algorithms, five state-of-the-art algorithms are used, simple DQN [29], DU-DRL with Random Provisioning (DU-RP), DU-DRL with Equally Distributed Provisioning (DU-EDP) [5], DU-DRL with Greedy Provisioning (DU-GP), and DQN only with Stage 1 (DQN-S1).

- DU-RP: virtual clusters of multi-tenant are random distributed during the fast initial provisioning stage, and then adaptively updated based on the DU-DRL.
- DU-EDP: virtual clusters of multi-tenant are equally distributed based on the algorithm proposed in [5] during the fast initial provisioning stage, and then adaptively updated based on the DU-DRL.
- DU-GP: virtual clusters of multi-tenant are greedy provisioned based on the capacities of physical machines in the DCN during the fast initial provisioning stage, and then adaptively updated based on the DU-DRL.
- DQN-S1: virtual clusters of multi-tenant are provisioned based on MFIP, and updated based on simple DQN.

5.2 Experiment results of baseline algorithms

We first conducted a thorough analysis of the first stage and verified its effectiveness in achieving optimal results within

its given constraints under the DCN with topologies $k = 6$, $k = 7$, $k = 8$, and $k = 9$. The results are shown in Figure. 4. In the first stage, we have the following observations: (i). The MFIP algorithm has the greatest elasticity value under different topological configurations in the first stage. Among the comparison algorithms, the GP algorithm has the worst performance, mainly because the greedy algorithm selects one of the computational and communication resources in each decision process without considering the bottlenecks and trade-offs. By comparison, the results of EDP and Random algorithms are relatively better than GP, while the EDP relies on the distribution of the remaining resources of the underlying physical servers, and the Random algorithm depends on the placement position of the servers selected by the randomized method each time slot. (ii). The MFIP algorithm has the maximum elasticity value in the first stage for various sets of user requests with the same topology. Since the physical resources are comparatively adequate when the scale of the data center is large ($k = 9$), different provisioning strategies will not have a significant impact on elasticity, and the advantage of MFIP on the elasticity is not particularly obvious when the request size is modest (60, 70). However, the limited multi-dimensional resources ($k = 6$) might result in significant differences in the elasticities across various solutions as the number of user requests increases (60 to 90).

Based on the results of the first stage, we evaluate the elasticity value of the second stage as follows. We deploy the algorithms on tree topology with 6 to 9-layers under algorithms (DQN, DU-EDP, DU-GP, DQN-S1, DUES) on each group of datasets and calculate the elasticities within 300 iterations. Among them, DU-RP, DU-EDP, and DU-GP are based on the first-stage Random, EDP, and GP methods, respectively. We conducted experiments on the elasticity values of data centers with different topologies ($k = 6$, $k = 7$, $k = 8$, and $k = 9$). The experiment results are shown in Figures 5 to 8, and we have the following observations: (i). For the same group of users, the elasticity values obtained by different algorithms are quite different. Figure. 5 (a) shows the elasticities under the five algorithms for the number of 60 tenants with the 6-layer DCN. The first three columns show the elasticities under the DQN, DU-EDP, and DU-GP, which are all negative values. It means that these three algorithms do not give an appropriate solution for the virtual clusters of these 60 tenants. (ii). The choice of strategy used for the fast provisioning has an important influence on the final elasticity value. As shown in Figure. 5 (a), DU-EDP and DU-GP are the strategies that adding EDP and GP fast provisioning based on DQN. The final elasticity values under these two strategies are lower than simple DQN. However, the elasticity under the DUES strategy is the highest. Therefore, we have that the choice of strategies using for fast provisioning is very important for the elasticity of the DCNs. For example, the strategy DU-RP does not perform well in the first stage when $k = 6$, but as k increases, the elasticity values show better results, and the trend in the second stage mirrors that of the first stage and relies on the results. In addition, the DU-GP strategy could cause some crucial resources (such as certain computing or communication resources) to become scarce in the first phase. Such resource shortages could intensify

in the second phase and lead to lower elasticity values. Based on the dependence between the second stage and the first stage, we can clearly see that the performance of the second stage is closely related to the resource allocation strategy of the first stage. Efficient fast provisioning strategies for the multi-tenant will make great improvements to the final result, on the contrary, inefficient ones will bring bad effects. (iii). The algorithm DUES has the highest elasticity values among different topological configurations in the second stage. Among the comparison algorithms, DU-EDP performs poorly when the topology is $k = 6$ and $k = 7$, but its performance improves when the topology is $k = 8$ and $k = 9$ as shown in Figures 7 and 8. This could be because the DU-EDP strategy can better balance resource allocation in larger topologies, thereby avoiding resource bottlenecks that occur in smaller topologies. The algorithm DU-GP performs consistently poorly because it prioritizes allocations based only on the elasticity values of the computational resources in each decision process, without considering bottlenecks and tradeoffs. The performance of the DU-RP algorithm varies significantly due to the randomness of the algorithm, leading to inconsistent results. DU-DRL has an efficient effect on improving the elasticity, especially when the occupancy of physical resources is not close to saturation. Comparing sub-figures (a) and (d) of Figure. 5, the difference between the last two columns of 60 tenants is much higher than 90. The average resource utilization of the DCNs with 60 and 90 tenants is nearly 80% and 98% under the DUES. Then, the number of tenants is large but not reaching saturation, and the impacts of algorithms on the elasticities are higher. (iv). The trends of a different group of multiple tenants with the same physical topology are various. Compared with sub-figures (a) to (d) of Figures. 5 to 8, we can see that a larger number of tenants in one group will lead to lower elasticity for the same physical topology. The reason is that more virtual clusters demand more physical resources, which will lead to an increase in the combinational utilization of the clusters and thus lower the elasticity of the data centers. Thus, a good provisioning scheme can support more virtual clusters of tenants in the larger data center network. In summary, compared with DQN, DU-EDP, DU-GP, and DQN-S1, DUES has better performance in elasticity across the virtual cluster provisioning in multi-tenant DCNs. The average elasticity can improve 1.91 times compared with DQN-S1 under the ranges of tenants $[60, 90]$.

5.3 Experiment results under different DCNs

Based on the experiment results of baseline algorithms under the 6-layer DCN, we conduct the experiments of DUES to compare the elasticities under different topologies. For each group of users, we collect five groups of results under the same settings.

5.3.1 Convergence

We investigate the convergence for the groups of tenants (60, 70, 80, and 90) under physical topologies with different layers (7-layer, 8-layer, and 9-layer). The results are shown in Figures. 9 to 11. The gray parts are the ranges of the collection results, and the bright lines with different colors are the
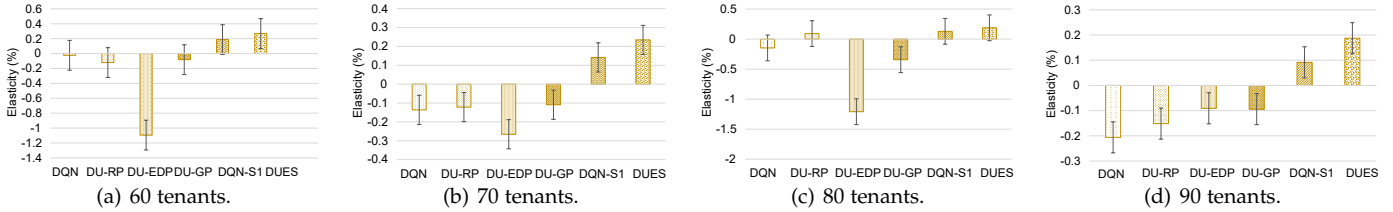
Fig. 5. The elasticities of 6-layer DCNs with tenants ranging from 60 to 90.
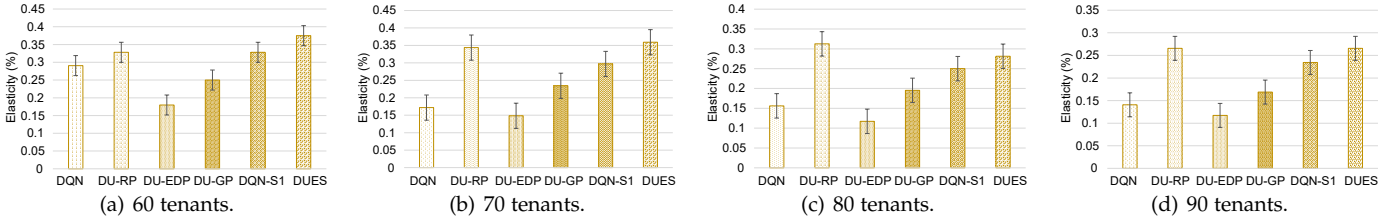

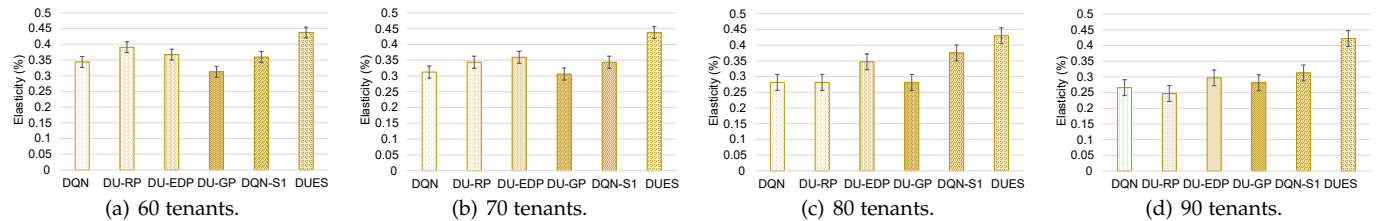Fig. 6. The elasticities of 7-layer DCNs with tenants ranging from 60 to 90.

Fig. 7. The elasticities of 8-layer DCNs with tenants ranging from 60 to 90.
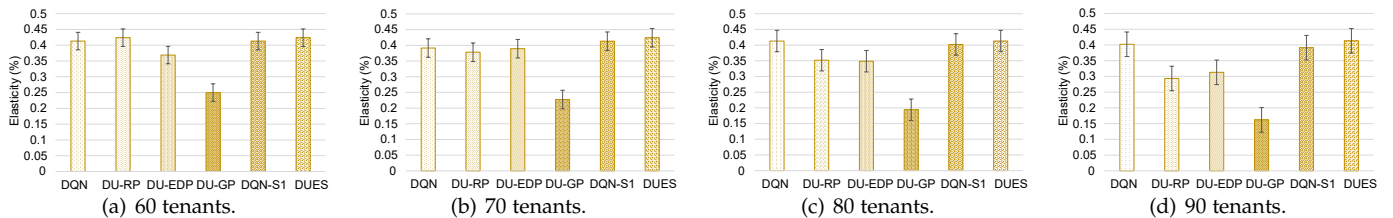

Fig. 8. The elasticities of 9-layer DCNs with tenants ranging from 60 to 90.

mean values. Additionally, we have the following observations: (i). The increasing number of tenants has an influence on the convergence. As shown in Figure. 9, the elasticity of the 7-layer data center is scaling with the increasing number of iterations. For each group of tenants, the elasticity begins to converge between 50 and 100 iterations and keeps at a high level of around 300 iterations. The growth rate is slow when the number of users is small, i.e. sub-figures (a) and (b) of Figure. 9, while it is relatively fast when the number of users is scaling. The reason is that the higher number of the total virtual clusters provisioning in the data center greatly decreases the rest of the available physical resources, which leads to a reduction of the searching space. (ii). The elasticity fluctuates within a relatively fixed range. There are many different placement results in the learning process of DUES, and the elasticities generated by these results will fluctuate among several relatively fixed values in the convergence process. As shown in the sub-figures (a), (b), and (c) of Figure. 9, the fluctuation of elasticities is within the range between 0.1 and 0.2. Compared with 8-layer and 9-layer, the fluctuation of elasticities is within the range of 0.1 and

0.12, respectively. This range is correlated to the topology of the DCN and the provisioning deviation of a few individual VMs.

5.3.2 Elasticity

According to the convergence of the elasticities obtained under the topologies with different layers, we assess the average of the highest elasticities among the five groups of results with ($\lambda = 0.5$, $\varphi = 0.5$) which are shown in Figure. 12. Additionally, we have the following observations: (i). The final elasticities decrease with the increasing number of tenants under the same t opology. As shown in sub-figure (a) of Figure. 12, the increasing rates between the number of tenants and the elasticities are similar, which is not the case in the other three groups. Such as the 7-layer topology group, the gap of the elasticity between 70 and 80 is not large, however, when the number of tenants increases to 90, the value of the elasticity decreases sharply. Since the sizes of the requested virtual clusters ranges in $[10, 20]$, the total amount of resources requested of the group with a lower number of tenants may be close to the higher one,
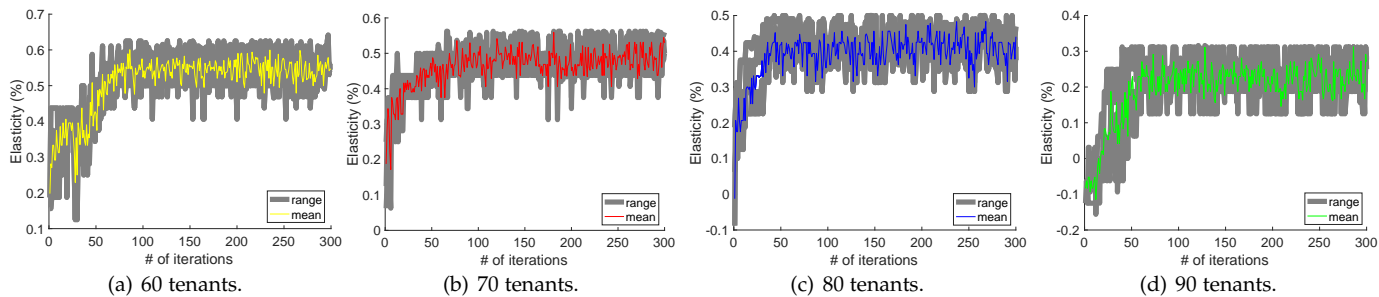
(a) 60 tenants. (b) 70 tenants. (c) 80 tenants. (d) 90 tenants.

Fig. 9. The elasticity under 7-layer topology.

(a) 60 tenants. (b) 70 tenants. (c) 80 tenants. (d) 90 tenants.
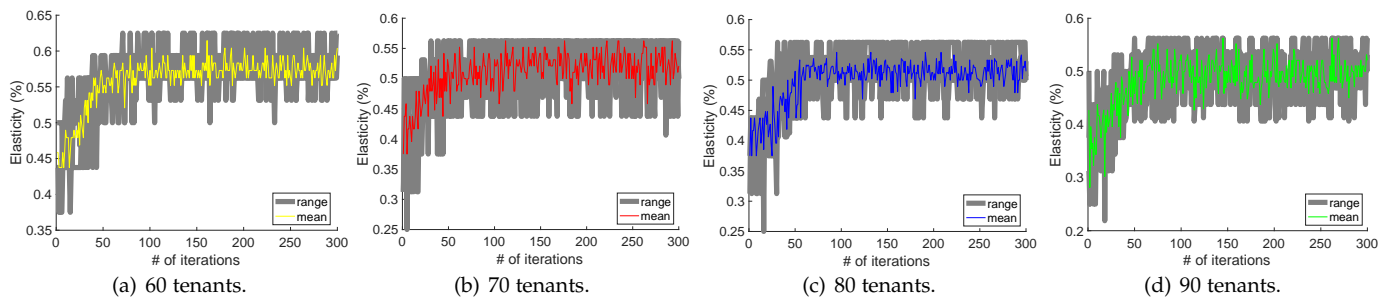
Fig. 10. The elasticity under 8-layer topology.

(a) 60 tenants. (b) 70 tenants. (c) 80 tenants. (d) 90 tenants.
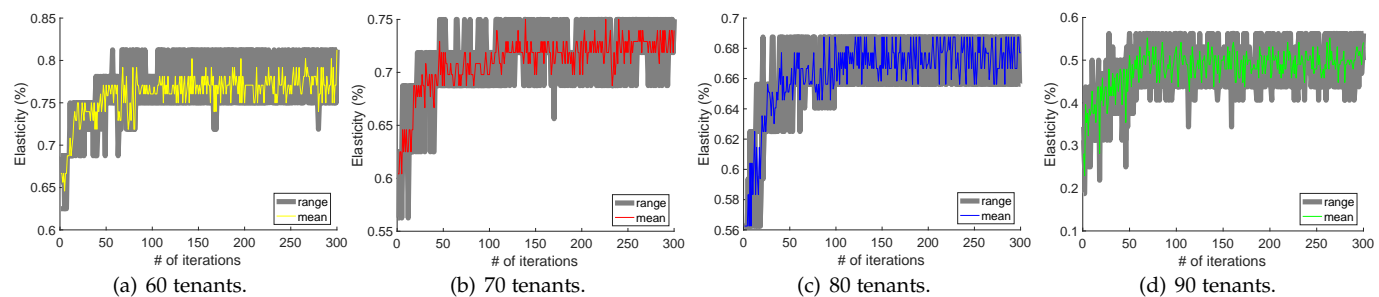
Fig. 11. The elasticity under 9-layer topology.

(a) 6-layer topology. (b) 7-layer topology. (c) 8-layer topology. (d) 9-layer topology.
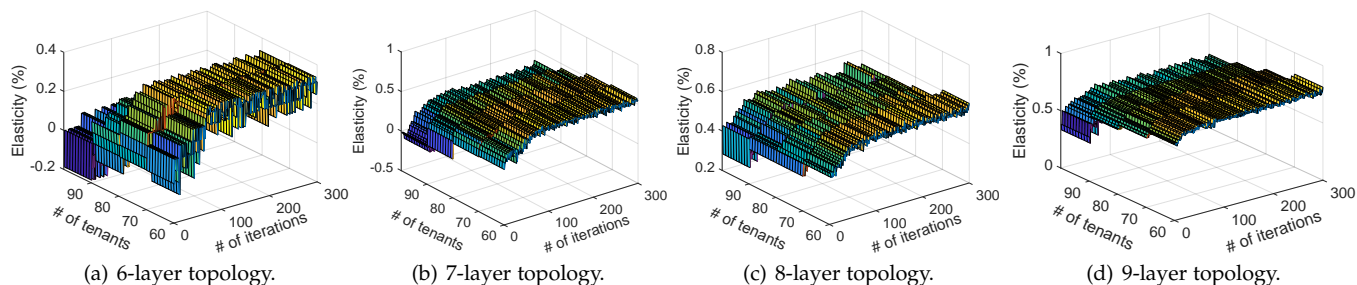
Fig. 12. The elasticities under different DCNs.

which leads to a not large gap of these groups. (ii). With the scaling of the topologies of the DCNs, the impacts of algorithms on the elasticities are higher. However, when the resources are sufficient to a certain level, the effect of updating strategy on elasticity may be small as shown in sub-figure (d) of 9-layer topology. Based on the experiment results of benchmarks under the 6-layer DCN, the elasticity depends on the localities of VMs requesting from different virtual clusters, which means a provisioning scheme can support more virtual clusters in a larger DCN. Compared with the same column on Y-axis with sub-figures (a) to (d)

of Figure. 12, the elasticities are increasing with the scaling topologies of the DCNs, which means more available resources can be supported by the providers. Additionally, we estimate the descent rates of several DCNs and compare the average elasticities; the results are shown in Figure. 13. We see that the fluctuations in elasticities decrease sharply when the scales of the topologies are 6 and 7 layers. However, the trend is flat when the scales of the topologies increase to 8 and 9 layers. Thus, we find that the fluctuation rates are inversely proportional to the scaling of the topologies. In summary, DUES in multi-tenant DCNs shows better
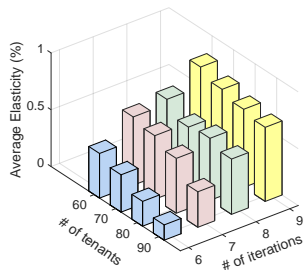
layers	descending rate
6	29.4
7	18.6
8	9.12
9	7.34

Fig. 13. The average elasticity and descending rate (%) of DCNs.

Fig. 14. The elasticities of the different scaling requirements in extreme scenarios.

TABLE 4
Coefficient Setting.

group	1	2	3	4	5
(λ,φ)	(0.3,0.7)	(0.4,0.6)	(0.5,0.5)	(0.6,0.4)	(0.7,0.3)

performance in terms of elasticity at different topologies.

5.3.3 Coefficient

In order to further analyze the impact of the coefficient on the combinational elasticity, we experimented on two distinct scenarios to confirm the efficacy of the introducing coefficients, and we used λ and φ to represent coefficients of physical machines and links, respectively. The setting of coefficients is shown in Table 4. We first evaluate the scenario where only compute resources are increased and users in the data center request an increase of 1 virtual machine. Due to the scaling of virtual machines between different virtual clusters, the communication between them is 0. Here, we assume that the current number of tenants in the data center is 90, which can be represented as (90, 1). The result is presented as follows. It is obvious that there are differences in the results of elasticity at different coefficients, and the result is shown in Figure. 14(a). Among them, the elasticity value is highest under the coefficient (0.4, 0.6), which indicates that the expansion of this group of queries is focused on physical machines and the subsequent scalability advantage is in communication resources. We then evaluate the scenario where only one virtual cluster has a scaling requirement. Since only the internal resources within one cluster have grown, the communication resources account for the main part where scaling in $[45, 90]$ Gbps based on the function $f(V_k, C_i)$. Here, we assume that the current cluster resource growth requirement is 90, which can be represented as (1, 90). The result is shown in Figure. 14(b). Among them, the elasticity value is the highest under the coefficient (0.3, 0.7), which indicates that the expansion of this group of queries is focused on physical links and the subsequent scalability advantage is in computing resources. Based on the above result, we have that the values of coefficients can more accurately reflect the various requirements for different resource categories and optimize the allocation by introducing coefficients to weigh the importance of E_m and E_l.

6 CONCLUSION

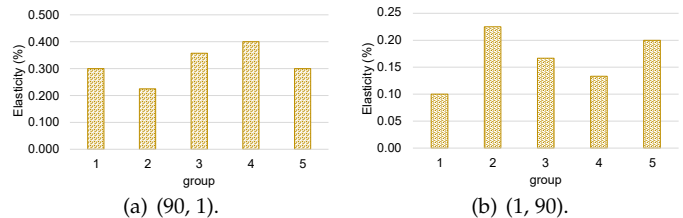In this paper, we address the virtual cluster provisioning problem in multi-tenant cloud data centers. We use elasticity to measure the potential growth of multi-tenant in terms of computing and communication resources. We aim to minimize the elasticity by designing a two-stage framework DUES, which consists of two stages. In the first stage, we first propose a fast initial provisioning MFIP scheme to realize the rapid response of multi-tenant, and we prove that MFIP is optimal under the single computation resource constrain. In the second stage, we propose a dynamic updating strategy DU-DRL based on deep reinforcement learning to further improve the elasticity of virtual clusters that are in use for scaling. Additionally, to avoid the high dimensions caused by the large scales of tenants and the DCN, we propose to train a fully connected neural network by designing a new feasible action set to realize the reduction, and it approximates the policy based on the proposed aggressive objective selection method in DU-DRL. Finally, we conduct extensive evaluations under various scenarios to demonstrate that our scheme outperforms existing state-of-the-art methods in terms of both elasticity and efficiency.

REFERENCES

[1] Hadary, O, Marshall, L, Menache, I, Pan, A, Greeff, E. E, Dion, D, and Moscibroda, T, "Protean:VM Allocation Service at Scale," In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2020, pp. 845-861.

[2] Yi, D, Zhou, X, Wen, Y, and Tan, R, "Efficient compute-intensive job allocation in data centers via deep reinforcement learning," IEEE Transactions on Parallel and Distributed Systems, vol 31(6), pp. 1474-1485, 2020.

[3] Lu, S, Wu, J, and Fang, Z, "High-elasticity Virtual Cluster Placement in Multi-tenant Cloud Data Centers," In IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019, pp. 996-1003.

[4] Wu, H, Zhang, W, Xu, Y, Xiang, H, Huang, T, Ding, H, and Zhang, Z, "Aladdin: optimized maximum flow management for shared production clusters," In 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2019, pp. 696-707.

[5] Wu, J, Lu, S, and Zheng, H, "On maximum elastic scheduling of virtual machines for cloud-based data center networks," In 2018 IEEE International Conference on Communications (ICC), 2018, pp. 1-6.

[6] Plummer, D. C, SMITH, D, BITTMAN, T, Cear-Ley, D. W, CAPPUCCIO, D, Scott, D, and Robertson, B, "Study: Five refining attributes of public and private cloud computing," Gartner, Tech Rep, 2009.

[7] Han, R, Ghanem, M. M, Guo, L, Guo, Y, and Osmond, M, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," Future Generation Computer Systems, vol 32, pp. 82-98, 2014.

[8] Foerster, K. T., and Schmid, S. "Survey of reconfigurable data center networks: Enablers, algorithms, complexity," ACM SIGACT News, vol 50(2), pp.62-79, 2019.

[9] Naskos, A, Stachtiari, E, Gounaris, A, Katsaros, P, Tsoumakos, D, Konstantinou, I, and Sioutas, S, "Dependable horizontal scaling based on probabilistic model checking," In 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 31-40.

[10] Farokhi, S, Jamshidi, P, Lakew, E. B, Brandic, I, and Elmroth, E, "A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach," Future Generation Computer Systems, vol 65, pp. 57-72, 2016.

[11] Lin, W., Li, X. Y., Chang, J. M., & Jia, X. (2023). "Constructing Multiple CISTs on BCube-Based Data Center Networks in the Occurrence of Switch Failures," IEEE Transactions on Computers.

[12] Fan, W., Xiao, F., Cai, H., Chen, X., & Yu, S. (2023). "Disjoint Paths Construction and Fault-tolerant Routing in BCube of Data Center Networks," IEEE Transactions on Computers.

[13] Wang, Y, Tan, C. C, and Mi, N, "Using elasticity to improve inline data deduplication storage systems," In 2014 IEEE 7th International Conference on Cloud Computing, 2014, pp. 785-792.

[14] Chowdhury, M, Liu, Z, Ghodsi, A, and Stoica, I, "HUG: Multi-resource fairness for correlated and elastic demands," In 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), 2016, pp. 407-424.

[15] de Alfonso, C, Caballer, M, Calatrava, A, Moltó, G, and Blanquer, I, "Multi-elastic datacenters: auto-scaled virtual clusters on energy-aware physical infrastructures," Journal of Grid Computing, vol 17(1), pp. 191-204, 2019.

[16] Kholidy, H. A, "An intelligent swarm based prediction approach for predicting cloud computing user resource needs," Computer Communications, vol 151, pp. 133-144, 2020.

[17] Guerrero, C, Lera, I, and Juiz, C, "Genetic algorithm for multi-objective optimization of container allocation in cloud architecture," Journal of Grid Computing, 16(1), pp. 113-135, 2018.

[18] Qaddoum, K. S, El Emam, N. N, and Abualhaj, M. A, "Elastic neural network method for load prediction in cloud computing grid," International Journal of Electrical and Computer Engineering, vol 9(2), pp. 1201, 2019.

[19] Wang, J., Zhao, G., Xu, H., Zhao, Y., Yang, X., & Huang, H. (2022, May). "TRUST: Real-Time Request Updating with Elastic Resource Provisioning in Clouds," IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, London, United Kingdom, pp. 620-629, 2022.

[20] Henderson, P, Islam, R, Bachman, P, Pineau, J, Precup, D, and Meger, D, "Deep reinforcement learning that matters," In Proceedings of the AAAI Conference on Artificial Intelligence, vol 32(1), 2018.

[21] Mao, H, Alizadeh, M, Menache, I, and Kandula, S, "Resource management with deep reinforcement learning," In Proceedings of the 15th ACM workshop on hot topics in networks, 2016, pp. 50-56.

[22] Bitsakos, C, Konstantinou, I, and Koziris, N, "Derp: A deep reinforcement learning cloud system for elastic resource provisioning." In 2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2018, pp. 21-29.

[23] Liang, S, Yang, Z, Jin, F, and Chen, Y. "Data centers job scheduling with deep reinforcement learning," In Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2020, pp. 906-917.

[24] Liu, K., Peng, J., Wang, J., Yu, B., Liao, Z., Huang, Z., & Pan, J. (2019). "A learning-based data placement framework for low latency in data center networks," IEEE Transactions on Cloud Computing, 10(1), 146-157.

[25] Nouri, S. M. R, Li, H, Venugopal, S, Guo, W, He, M, and Tian, W, "Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications," Future Generation Computer Systems, vol 94, pp. 765-780, 2019.

[26] Chen, L, Lingys, J, Chen, K, and Liu, F, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," In Proceedings of the 2018 conference of the ACM special interest group on data communication, 2018, pp. 191-205.

[27] Ying, C, Li, B, Ke, X, and Guo, L, "Raven: Scheduling Virtual Machine Migration During Datacenter Upgrades with Reinforcement Learning," Mobile Networks and Applications, pp. 1-12, 2020.

[28] Lu, S. B, Wu, J, Zheng, H. Y, and Fang, Z. Y, "On maximum elastic scheduling in cloud-based data center networks for virtual machines with the hose model," Journal of Computer Science and Technology, vol 34(1), pp. 185-206, 2019.

[29] Mnih, V, Kavukcuoglu, K, Silver, D, Rusu, A. A, Veness, J, Bellemare, M. G, and Hassabis, D, "Human-level control through deep reinforcement learning," nature, vol. 518(7540), pp. 529-533, 2015.

[30] Gao, C., Chu, S., Xu, H., Xu, M., Ye, K., & Xu, C. Z. (2021). "Flash: Joint flow scheduling and congestion control in data center networks," IEEE Transactions on Cloud Computing.

[31] Tran, V., Sun, J., Tang, B., & Pan, D. (2022, May). "Traffic-optimal virtual network function placement and migration in dynamic cloud data centers," In 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (pp. 919-929). IEEE.

[32] Fei, B, Zhu, X, Liu, D, Chen, J, Bao, W, and Liu, L. "Elastic resource provisioning using data clustering in cloud service platform," IEEE Transactions on Services Computing, 2020.

[33] Yazdanbakhsh, M, Isfahani, R. K. M, and Ramezanpour, M, "MODE: a multi-objective strategy for dynamic task scheduling through elastic Cloud resources," Majlesi Journal of Electrical Engineering, vol. 14(2), pp. 127-141, 2020.

[34] Bao, Y, Peng, Y, and Wu, C, "Deep learning-based job placement in distributed machine learning clusters," In IEEE INFOCOM 2019-IEEE conference on computer communications, 2019, pp. 505-513.

[35] Arulkumaran, K, Deisenroth, M. P, Brundage, M, and Bharath, A. A, "Deep reinforcement learning: A brief survey," IEEE Signal Processing Magazine, 34(6), pp. 16-38, 2017.

Shuaibing Lu is currently a lecturer of Faculty of Information Technology of Beijing University of Technology. She received her Ph.D. degree in Computer Science and Technology from Jilin University, Changchun, in 2019. She is supported by the China Scholarship Council as a visiting scholar supervised by Prof. Jie Wu in the Department of Computer and Information Sciences at Temple University (2016-2018). She is a member of IEEE. Her current research focuses on distributed computing, cloud computing and fog computing.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was 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. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Jiamei Shi received her B.Sc. in Computer Science and Technology of Shanghai Maritime University. Currently, she is working toward a M.Sc degree on Computer Science with Faculty of Information Technology of Beijing University of Technology. Her research interests include cloud computing and edge computing.

Juan Fang Juan Fang, received the M.S. degree from Jilin University of Technology, Changchun, China in 1997, and her Ph.D. degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology. From 2015, she is the professor of Beijing University of Technology. Her research interests include high performance computing, edge computing and big data technology.

Jiayue Zhang is currently a lecturer of the Faculty of Information Technology of Beijing University of Technology. She received her Ph.D. in Signal and Information Processing from Beijing University of Posts and Telecommunications, Beijing, in 2018. She is supported by the China Scholarship Council as a visiting scholar supervised by Prof. Jimmy Huang in the Department of Information Technology at York University (2012-2014). She is a member of IEEE and China Comupter Federation (CCF). Her current research focuses on the Machine Learning and Data Mining.

Haiming Liu Haiming Liu is currently a lecturer of School of Software Engineering at Beijing Jiaotong University. He received his Ph.D. degree in Computer Science and Technology (Bioinformatics) from Jilin University, Changchun, in 2019. Before that, he received his M.S. degree in Computer Software and Theory from Jilin University, Changchun, in 2015 and B.S. degree in Computer Science and Technology from Jilin University, Changchun, in 2012. He is a member of Chinese Association for Artificial Intelligence (CAAI). His current research focuses on the edge computing, data mining, and bioinformatics.