CADER: Cost-Efficient Cloud Application Deployment With Tenant Requirement Guarantee in Multi-Clouds

Huaqing Tu, *Member, IEEE*, Ziqiang Hua, Qianpiao Ma, Hanguang Luo, Tao Zou, Gongming Zhao, *Member, IEEE*, and Hongli Xu, *Member, IEEE*

***Abstract*—Motivated by the need to reduce vendor lock-in and address concerns regarding dedicated hardware availability, cloud applications have increasingly adopted a multi-cloud deployment strategy, in which cloud applications are deployed in different zones associated with various cloud service providers. When deploying cloud applications in multi-clouds, there are three crucial and coupled metrics: *deployment cost*, *access delay* and *traffic demand*. Unfortunately, existing works overlook either the data transfer cost in deployment cost or the access delay and traffic demand requirements, resulting in high operating costs or low user QoS. To bridge this gap, this paper proposes the Cost-Efficient Application Deployment Framework (CADER) with tenant requirement guarantee in multi-clouds environment. However, due to the challenges of service price heterogeneity, transfer cost diversity, and resource limitation, achieving cost-efficient cloud application deployment while satisfying all tenant requirements is not an easy task. To tackle this issue, we design an approximate algorithm based on the random rounding method and prove that its approximate ratio is $O(\log g)$, where g is the number of cloud zones. Results of in-depth simulations indicate that CADER can reduce the application deployment cost ranging from 16% to 38% compared to commonly used alternatives while ensuring the satisfaction of tenant requirements.**

***Index Terms*—Cloud computing, multi-clouds, cloud application deployment, task placement, cost effectiveness.**

Received 3 April 2024; revised 14 October 2025; accepted 26 October 2025. Date of publication 28 October 2025; date of current version 8 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grants U22A2005 and Grant 62402537, in part by the Fundamental Research Funds for the Central Universities under Grant 30925010408, and in part by China Postdoctoral Science Foundation under Grant 2024M75986. Recommended for acceptance by M. Govindaraju. *(Corresponding authors: Qianpiao Ma; Gongming Zhao.)*

Huaqing Tu is with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China, and also with Zhejiang Lab, Hangzhou, Zhejiang 310000, China (e-mail: thq@mail.zjgs.edu.cn).

Ziqiang Hua, Hanguang Luo, and Tao Zou are with Zhejiang Lab, Hangzhou, Zhejiang 310000, China (e-mail: ziqhua@zhejianglab.com; luohg@zhejianglab.com; zout@zhejianglab.com).

Qianpiao Ma is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China (e-mail: maqianpiao@pmlabs.com.cn).

Gongming Zhao and Hongli Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China, and also with the Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu 215123, China (e-mail: gmzhao@ustc.edu.cn; xuhongli@ustc.edu.cn).

Digital Object Identifier 10.1109/TCC.2025.3626582

I. Introduction

NOWADAYS, it is increasingly common that cloud applications (*e.g.*, Disney+ [1], Netflix [2]) are deployed across multiple cloud providers (multi-cloud) instead of relying solely on one cloud service provider. This shift is driven, in part, by factors such as privacy regulations, the availability of specialized hardware, the desire to avoid vendor lock-in, and the optimal latency and load times for customers [3], [4], [5]. According to the recent survey [6], 96% reported that they are using or plan to use at least two cloud service providers with 45% using cloud applications from five or more providers. Thus, it is increasingly important to support the cloud application deployment in multi-clouds. A cloud application is typically comprised of multiple tasks and it is common for a cloud service provider to offer a range of cloud zones. In this paper, we concentrate on the problem of placing tasks of a cloud application in cloud zones across multiple cloud service providers. As illustrated in Fig. 1, cloud zones offered by multiple cloud service providers are dispersed across various geographical locations, providing global coverage and geographical options for cloud application deployment to cloud users, also known as tenants.

When placing cloud applications in multi-clouds, deployment cost, access delay and traffic demand are the three important metrics from the perspective of tenants.

- *Deployment cost* refers to the total expenses associated with placing cloud applications in multi-clouds. It consists of two components: the cost of service acquisition (*e.g.*, purchasing VMs) and the cost of data transfer (*e.g.*, sending data from one cloud zone to another cloud zone) [4]. The tenants expect to minimize the deployment costs as much as possible.
- *Access delay* refers to the latency incurred during round-trip access between users and the cloud-deployed services. It is typically a pivotal factor that tenants take into account when deploying applications, as excessive delay can result in a diminished user experience [7].
- *Traffic demand:* A cloud application is mainly composed of a number of tasks. Traffic demand pertains to the necessary communication delay and bandwidth requirements for multiple tasks within the same cloud application. Failure to consider traffic demand can lead to prolonged task completion times.

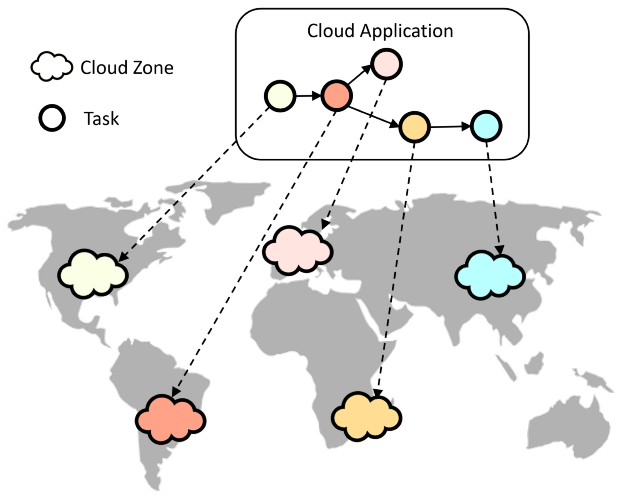


Fig. 1. An example of deploying cloud applications across clouds. The cloud application consists of five tasks. Different colored cloud zones indicate that they belong to different cloud service providers. Tasks within the cloud application can be placed in different cloud zones.

TABLE I
SERVICE PRICE HETEROGENEITY. DATA IS COLLECTED BY [4]. THERE ARE 294 ZONES GLOBALLY ACROSS THE TOP 3 CLOUDS (GCP, AWS AND AZURE). "MAX/MIN" REFERS TO THE PRICE RATIOS BETWEEN THE MOST EXPENSIVE AND THE LEAST EXPENSIVE ZONE. "CV" INDICATES THE COEFFICIENTS OF VARIATION (CV) OF PRICES ACROSS DIFFERENT ZONES.

Type	Hardware	Zones	Max/Min	CV
CPU	AMD (8 cores)	146	2.5×	16%
	Intel (8 cores)	248	1.6×	12%
GPU	T4 (1 chip)	146	1.7×	12%
	K80 (1 chip)	56	9.5×	48%
TPU	v2 (8 cores)	5	1.2×	6%
	v3 (8 cores)	4	1.1×	4%

Tenants expect to minimize deployment costs while meeting the access delay and traffic demand requirements. The intra-zone deployment solutions can not be applied in multi-clouds as they fail to meet the three important tenant requirements mentioned above, necessitating a novel design approach for cost-efficient cloud deployment in multi-clouds. In pursuit of this ambitious objective, this paper seeks to tackle a significant and unexplored question: Can cloud applications be deployed with minimal deployment costs in multi-clouds while satisfying access delay and traffic demand requirements? However, achieving such cost-efficient cloud application deployment is a challenging mission due to the interplay of three interconnected factors, namely, service price heterogeneity, transfer cost diversity and zone-related resource limitation.

- *Service price heterogeneity:* Typically, cloud service providers deploy cloud infrastructure across multiple regions, with each region comprising several availability zones [7]. Table I shows that even for the same service, there is a significant price difference in different zones. For example, TPUs are offered in only 4-5 (or 5%) GCP zones, yet there is still a 10% -20% price difference across those zones [4].
- *Transfer cost diversity:* Cloud service providers levy charges for outbound data transfer from a zone. Table II shows the data transfer cost of inter-zone, inter-region and between continents across the top 3 clouds. We can observe significant variations in pricing among different cloud service providers.Hence, the choice of cloud service provider and the selection of the specific region within that provider for cloud application deployment have a significant impact on the cost of data transfer.
- *Zone-related resource limitation* encompasses the availability of resources (*e.g.*, computing, storage) that a zone can provide to tenants, as well as the bandwidth constraints for communication between zones [8], [9]. The constrained zone-related resource and bandwidth capacity among zones may pose challenges when tasks require significant resources and involve a substantial volume of traffic.

TABLE II
TRANSFER COST DIVERSITY [10]. IT SHOWS THE DATA TRANSFER COST OF INTER-ZONE, INTER-REGION AND BETWEEN CONTINENTS ACROSS THE TOP 3 CLOUDS.

Traffic Type	GCP	AWS	Azure
Traffic (inter-zone)	0.01\$/GB	0.01\$/GB	0.01\$/GB
Traffic (inter-region) US	0.01\$/GB	0.01\$/GB	0.02\$/GB
Traffic (inter-region) ASIA	0.05\$/GB	0.01\$/GB	0.08\$/GB
Traffic (between continents)	0.08\$/GB	0.02\$/GB	0.02\$/GB

Given a cloud application with access delay and traffic demand requirements, the three coupled factors discussed above make it hard for tenants to find the best cloud application deployment scheme with minimal deployment cost in multi-clouds. There is an urgent need to find a practical, efficient and all-in-one cloud application deployment solution. To this end, this paper proposes the Cost-Efficient Application Deployment Framework (CADER), wherein the requirement of access delay and traffic demand specified by tenants are thoroughly taken into consideration and a cloud application can be deployed cost-effectively. The cornerstone of CADER is an efficient approximate algorithm that minimizes the overall deployment cost while satisfying all tenant requirements and adhering to resource constraints. In brief, this research makes the following contributions:

1) We put forth an optimization framework called CADER, which encompasses a mathematical model of the cloud application deployment problem in multi-clouds. It aims at minimizing overall deployment costs while satisfying all diverse tenant requirements and resource limitations.
2) We define the deployment problem as a non-linear optimization problem. However, it is hard to solve since it is a complex convex problem. To resolve it, we introduce auxiliary variables and then design an efficient algorithm. Through extensive analysis, we prove that the approximate

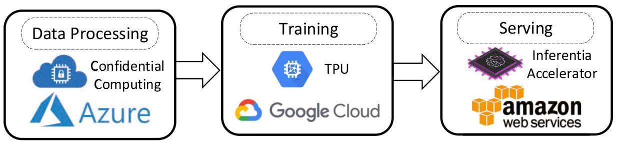


Fig. 2. A example of multi-cloud deployment strategy. An ML pipeline consists of data processing, training and serving, which are respectively deployed on Azure, GCP, and AWS to leverage the advantages offered by various cloud service providers and ensure secure data processing.

ratio of our proposed algorithm is $O(\log g)$, where g is the number of cloud zones.

3) We perform comprehensive simulations using real-world cloud zone topology, service and data transfer price, and cloud application datasets, *i.e.*, Google Cluster Trace [11]. The results demonstrate that CADER achieves a reduction in deployment costs of up to 16% -38% when compared to existing solutions.

II. Background and Related Work

A. A Brief of Multi-Cloud Deployment

Multi-cloud deployment refers to the practice of distributing and running cloud applications or services across multiple cloud computing providers. It involves leveraging the capabilities and resources of different cloud zones simultaneously to achieve specific goals. This approach allows tenants to take advantage of the unique features, services, and pricing models offered by different cloud service providers. By spreading workloads across multiple clouds, tenants can enhance flexibility, scalability, and resilience. The benefits of multi-cloud deployment include avoiding vendor lock-in, optimizing costs by leveraging competitive pricing, improving performance by leveraging specialized services, and enhancing data redundancy and disaster recovery capabilities. It also provides the ability to select the most suitable cloud zone for each specific workload or application. Fig. 2 illustrates an example of the multi-cloud deployment strategy, taking machine learning (ML) training as an instance. The ML pipeline comprises three tasks: data processing, training, and serving. The cloud customers, aka tenants, wish to leverage the advantages offered by various cloud service providers (*e.g.*, AWS [12], Azure [13], GCP [14], *etc.*) to the fullest extent while ensuring secure data processing. They can opt to utilize Azure Confidential Computing [15] for data processing to ensure data anonymization and confidentiality, leverage GCP for training purposes to benefit from TPUs [16], and employ AWS for serving to take advantage of the Inferentia Accelerator [17]. According to the recent surveys [18], [19], more than 86% of the 727 respondents have embraced the multi-cloud strategy to cater to diverse workloads. Thus, it is increasingly important to support the cloud application deployment in multi-clouds.

In a multi-cloud environment, cloud brokers [20] serve as intermediaries connecting cloud service providers and customers (*i.e.*, tenants). Their core role is to negotiate between all providers and cloud customers, identifying the most suitable services that balance customer preferences and provider profits. Currently, several existing programs and management frameworks function as cloud brokers. Terraform [21], an infrastructure-as-code tool, can be used for automated management of infrastructure across multiple cloud platforms, but it only focuses on resource management and does not handle cloud application deployment. Libcloud [22], JClouds [23], and RightScale [24] provide unified interfaces and management capabilities for the computing, storage, and other services of many cloud service providers. However, the user still needs to deploy cloud applications manually.

A cloud service provider typically has multiple regions worldwide to meet the needs of different markets. For example, AWS has approximately 30 regions globally, including US East (N. Virginia), US West (Oregon), and Asia Pacific (Mumbai), *etc.* Each region consists of multiple zones. For instance, in the US East (N. Virginia) region, AWS provides multiple zones such as us-east-1a, us-east-1b, us-east-1c, and so on. The services offered may vary between zones, and even the same service may have differences across different zones. When deploying cloud applications in multi-clouds, tenants need to carefully select the appropriate zones from different cloud service providers for each task to deploy their services to achieve the best service experience and minimize deployment costs as much as possible.

B. Related Work

Based on the deployment scenarios, we categorize existing works related to cloud application deployment into the following three types. The first two categories belong to intra-cloud deployment and the third one is inter-cloud deployment. Table III summarizes the strengths of CADER compared with the existing works.

Intra-Zone: There is a significant amount of research focused on how to deploy tasks within a specific cloud zone [25], [26], [27], [28], [36], [37]. Specifically, Tu et al. [25] propose a tenant-grained task placement method to reduce the scheduling delay and enhance system robustness. The authors have designed efficient algorithms with approximate ratio and competitive ratio guarantees for both offline and online scenarios. Gao et al. [27] use the reinforcement learning method and linear programming method to solve the problem of minimizing the number of SLO violations when allocating resources for tasks. Zhu et al. [36] propose a new task scheduling system, called WorkloadCompactor, which can minimize data center resource costs while meeting tail latency Service Level Objectives (SLOs). Li et al. [37] present TrafficShaper, which can shape the traffic between tasks so as to reduce or even minimize the transmission cost. However, these algorithms are only used when the task has been known to be executed in a specific zone, so they are not aware of cloud characteristics and tenant requirements. Thus, they can not be used to distribute tasks to multiple cloud zones.

Inter-Zone: Several prior studies [8], [9], [29], [30], [31] have focused on the cloud application deployment across zone within a cloud. Singh et al. [8] optimize the inter-zone bandwidth costs by leveraging the abundance of latency-equivalent peer links on the cloud zone edge without impacting latency significantly. Luo et al. [9] introduce TanGo, a cost optimization framework aimed at minimizing overall electricity expenses.

TABLE III
COMPARISON OF STRENGTHS AND WEAKNESS OF EXISTING WORKS

Deployment Solutions		Cloud Characteristics			Tenant Requirements		
		Service Price Heterogeneity	Transfer Cost Diversity	Zone-Related Resource Limitation	Placement Cost	Access Delay	Traffic Demand
Intra-Cloud	Intra-Zone (*e.g.*, [25]–[28])	✘	✘	✘	✘	✘	✘
	Inter-Zone (*e.g.*, [8], [9], [29]–[31])	*partial*	✘	✔	✘	*partial*	*partial*
Multi-Clouds (*e.g.*, [4], [32]–[35])		✔	✔	*partial*	✔	✘	✘
CADER		✔	✔	✔	✔	✔	✔

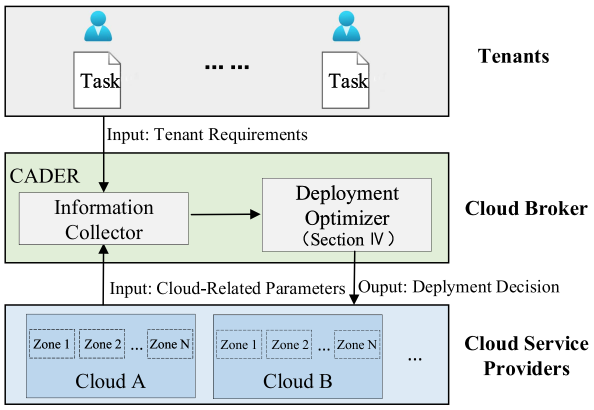


Fig. 3. Overview of CADER.

Employing the submodular-based technique, they craft a near-optimal algorithm that achieves a tight approximation ratio. Sarkar et al. [30] study a deadline-aware dynamic task placement strategy to offload and place the tasks in a suitable cloud zone. To overcome the limited scalability of application placement algorithms, Mann et al. [31] study a decentralized application placement algorithm, which has been proven to have better scalability compared to centralized application placement algorithms. Nevertheless, these works fail to account for the price heterogeneity of services offered by cloud service providers and differences in transfer cost between different zones, resulting in high placement cost of cloud application.

Multi-Clouds: Some previous works [4], [32], [33], [34], [35] study how to deploy cloud applications across clouds. Specifically, Yang et al. [4] present SkyPilot, an intercloud broker, to enable the deployment of cloud applications across clouds and support seamless task migration. It allows tenants to view the cloud ecosystem not just as a collection of individual clouds but as a more integrated Sky of Computing. Radulescu et al. [33] focus on how to take cost, evaluation risk and service performance into consideration when selecting appropriate cloud service providers. Zhao et al. [32] introduce an innovative satellite data processing system, which optimizes the transmission and processing efficiency of satellite data by integrating resources from multiple cloud service providers, effectively reducing costs and increasing the speed of data processing. Singh et al. [34] present an energy-efficient application deployment algorithm for multi-cloud environments. This algorithm can quickly generate deployment plans while meeting the response time constraints, effectively reducing system power consumption. Aldailamy et al. [35] focus on the optimization of data storage costs in online social networks in multi-clouds, and propose an algorithm framework based on dynamic replication and intelligent placement. By accurately predicting the access frequency of data objects and flexibly adjusting the distribution of replicas, it achieves dual optimization of cost and latency. However, these studies ignore the access delay for tenants and traffic demand among tasks within a cloud application, leading to low user QoS.

III. Network Model and Problem Formulation

A. Overview of CADER

Fig. 3 illustrates the overview of CADER. As an inter-cloud broker, CADER comprises two fundamental components: the Information Collector and the Deployment Optimizer. The Information Collector aggregates two types of inputs: 1) Tenant Requirements: This includes a directed acyclic graph (DAG) of tasks, access delay thresholds, service type requirements, resource demands (*e.g.*, CPU/GPU units, memory) and security/compliance requirements. These specifications can be explicitly provided by tenants. 2) Cloud-Related Parameters: This encompasses per-zone resource capacities, access delays, inter-zone bandwidth availability, service price, data transfer costs, per-zone compliance certification metadata (*e.g.*, HIPAA, PCI-DSS, GDPR certification status) and data residency tags (*e.g.*, EU Sovereign Cloud, Local Data Center). Such information can be obtained through cloud provider APIs (*e.g.*, AWS EC2 API, AWS Pricing API, AWS Artifact, Azure Compliance Manager) and Third-Party Aggregators (*e.g.*, CloudHarmony).

The collected data is then transmitted to the Deployment Optimizer, which executes the cloud application deployment algorithm (detailed in Section IV). This module generates optimized deployment decisions to ensure minimal cost while satisfying all tenant-specified constraints.

B. System Model

Cloud Model: In multi-clouds, there is a set of cloud service providers, which is denoted by $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, where $m = |\mathcal{C}|$ is the number of clouds. Each cloud provider has a set of cloud zones, which are distributed in different regions. The set of cloud zones belonging to cloud service providers $c \in \mathcal{C}$ is

TABLE IV
KEY NOTATIONS

Symbol	Description
$\mathcal{C}$	Set of cloud service providers
$\mathcal{R}_c$	Set of cloud zones belonging to $c \in \mathcal{C}$
$\mathcal{R}$	Set of zones from all cloud service providers
$\mathcal{T}$	Set of tasks within a cloud application
h_r	Resource capacity of zone $r \in \mathcal{R}$
$\tau_{r,r'}$	Communication delay between zones r and r'
τ_r	Access delay between tenant and zone $r \in \mathcal{R}$
$\mu_{r,r'}$	Available bandwidth resource between zones r and r'
α_r^t	Cost of placing task t in zone r
$\beta_{r,r'}$	Data transfer cost per traffic unit between zones r and r'
I_r^t	Binary constant indicating whether zone r supports service type t or not
δ_t	Access delay threshold for task $t \in \mathcal{T}$
φ_t	Resource demand of task $t \in \mathcal{T}$
$\delta_{t,t'}$	Communication delay threshold between tasks t and t'
$b_{t,t'}$	Bandwidth demand between tasks t and t'
x_r^t	Decision binary variable indicating whether task t is placed in zone r or not

denoted as $\mathcal{R}_c$ and cloud zones from all cloud service providers are represented as $\mathcal{R} = \mathcal{R}_{c_1} \cup \mathcal{R}_{c_2} \cup \ldots \cup \mathcal{R}_{c_m}$. The resource capacity, including computing and storage, *etc*, of zone $r \in \mathcal{R}$ is denoted as h_r. The communication delay, available bandwidth resource and data transfer cost per unit traffic between zones r and r' are denoted as $\tau_{r,r'}$, $\mu_{r,r'}$ and $\beta_{r,r'}$, respectively. We use h_r to denote the total resource capacity of region r. τ_r represents the access delay between tenant and zone r. The traffic delay and bandwidth capacity between region r and r' are denoted as $\tau_{r,r'}$ and $\mu_{r,r'}$, respectively. The existing techniques [38], [39] can be used to measure the delay and bandwidth between different zones.

Cloud Application Model: Each cloud application consists of multiple tasks. The task set is denoted as $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, where $n = |\mathcal{T}|$ is the number of tasks. One task may send data to another task, so each task set can be represented as a directed acyclic graph $\mathcal{G}$, where vertices represent tasks and edges represent dependencies between tasks. The access delay threshold and the resource requirement of task $t \in \mathcal{T}$ posed by the tenant are denoted as δ_t and φ_t, respectively. The traffic delay and bandwidth resource requirement between tasks t and t' are denoted as $\delta_{t,t'}$ and $b_{t,t'}$, respectively. The key notations used in this paper are summarized in Table IV.

C. Problem Formulation

Problem Statement: Given a cloud application, CADER aims to find an appropriate cloud zone for each task so that the deployment cost of cloud application are minimized, while the performance requirement posed by the tenant are satisfied. The inputs of CADER are tenant requirements and cloud-related parameters (detailed in Section III-A). The output is a set of binary decision variables, $\{x_r^t\}$, which represent the mapping between tasks and zones. If task t is placed in zone r, $x_r^t = 1$; otherwise, $x_r^t = 0$.

Constraints: When placing tasks in multiple clouds, the following constraints should be satisfied:

1) *Task Placement Constraint:* For any task $t \in \mathcal{T}$, it should be allocated to a single zone, without any overlapping or multiple assignments. It means $\sum_{r\in\mathcal{R}} x_r^t = 1, \forall t \in \mathcal{T}$.
2) *Service Type Constraint:* We use I_r^t to represent service type support and security/compliance requirement satisfaction. $I_r^t = 1$ only if zone r both provides the required service for task t and meets the tenant's security/compliance standards; otherwise, $I_r^t = 0$. Task $t \in \mathcal{T}$ can be place in zone $r \in \mathcal{R}$ if and only if zone r can provide the services required by task t. That is, $x_r^t \le I_r^t, \forall t \in \mathcal{T}, r \in \mathcal{R}$.
3) *Access Delay Constraint:* Tenants may have different access delay requirements for their tasks. Therefore, it is necessary that each task is placed in a zone that can satisfy its access delay constraint. It follows $x_r^t \cdot \tau_r \le \delta_t, \forall t \in \mathcal{T}, r \in \mathcal{R}$.
4) *Communication Delay Constraint:* There is a traffic delay when a task sends data to another task. The inter-task communication delay must not surpass the traffic delay threshold specified by the tenant. That is, $x_r^t \cdot x_{r'}^{t'} \cdot \tau_{r,r'} \le \delta_{t,t'}, \forall \langle t,t'\rangle \in \mathcal{T}, \langle r,r'\rangle \in \mathcal{R}$.
5) *Inter-Zone Bandwidth Constraint:* The available bandwidth resource between any pair of zones should satisfy the bandwidth resource requirement posed by tasks. It means that $\sum_{t,t'\in\mathcal{R}} x_r^t \cdot x_{r'}^{t'} \cdot b_{t,t'} \le \mu_{r,r'}, \forall \langle r,r'\rangle \in \mathcal{R}$.
6) *Zone Resource Constraint:* The resources, such as computing and storage, provided by a zone should satisfy the resource requirement of the task. It follows $\sum_{t\in\mathcal{T}} x_r^t \cdot \varphi_t \le h_r, \forall r \in \mathcal{R}$.

Objective: The objective of CADER is to minimize the total deployment cost, including service acquisition cost and data transfer cost, while ensuring compliance with tenant requirements and zone resource limitations. Let α_r^t be the cost of service acquisition for task t in zone r. The task placement cost can be represented as $\sum_{t\in\mathcal{T}} \sum_{r\in\mathcal{R}} x_r^t \cdot \alpha_r^t$. The total data transfer cost can be calculated according to $\sum_{r,r'\in\mathcal{R}} x_r^t \cdot x_{r'}^{t'} \cdot \beta_{r,r'} \cdot b_{t,t'}$. Minimizing the sum of these two parts is our optimization goal. We formalize the problem of cost-efficient cloud application deployment with tenant requirements guarantee (CAP) as follows:

$$\min \sum_{t\in\mathcal{T}} \sum_{r\in\mathcal{R}} x_r^t \cdot \alpha_r^t + \sum_{r,r'\in\mathcal{R}} x_r^t \cdot x_{r'}^{t'} \cdot \beta_{r,r'} \cdot b_{t,t'}$$
$$S.t. \begin{cases} \sum_{r\in\mathcal{R}} x_r^t = 1, & \forall t \in \mathcal{T} \\ x_r^t \le I_r^t, & \forall t \in \mathcal{T}, r \in \mathcal{R} \\ x_r^t \cdot \tau_r \le \delta_t, & \forall t \in \mathcal{T}, r \in \mathcal{R} \\ x_r^t \cdot x_{r'}^{t'} \cdot \tau_{r,r'} \le \delta_{t,t'}, & \forall t,t' \in \mathcal{T}, r,r' \in \mathcal{R} \\ \sum_{t,t'\in\mathcal{R}} x_r^t \cdot x_{r'}^{t'} \cdot b_{t,t'} \le \mu_{r,r'}, & \forall r,r' \in \mathcal{R} \\ \sum_{t\in\mathcal{T}} x_r^t \cdot \varphi_t \le h_r, & \forall r \in \mathcal{R} \\ x_r^t \in \{0,1\} & \forall t \in \mathcal{T}, r \in \mathcal{R} \end{cases} \tag{1}$$

The first set of constraints represents the task placement constraint. The constraints from the second to the sixth denote the type constraint, access delay constraint, communication delay constraint, inter-zone bandwidth constraint and zone resource constraint, respectively. Our objective is to minimize the overall deployment cost, *i.e.*, $\min \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}} x_r^t \cdot \alpha_r^t + \sum_{r,r'\in\mathcal{R}} x_r^t \cdot x_{r'}^{t'} \cdot \beta_{r,r'} \cdot b_{t,t'}$.

IV. Algorithm Design

A. Preliminaries

The CAP problem defined in (1) is hard to solve due to the following reasons: 1) there are binary variables $\{x_r^t\}$; 2) there are products of variables. Both of these two issues make the CAP problem defined in (1) a complicated non-convex optimization problem. Therefore, designing an efficient algorithm with approximation guarantee is non-trivial. Hereafter, we will show that how to convert (1) into a linear programming problem.

We introduce auxiliary variables $\{y_{r,r'}^{t,t'}\}$, which indicates if task t will be placed in zone r and task t' will be placed in zone r'. According to the definition of variables $\{x_r^t\}$ and $\{y_{r,r'}^{t,t'}\}$, we have:

$$y_{r,r'}^{t,t'} = x_r^t \cdot x_{r'}^{t'} \tag{2}$$

If we replace $x_r^t \cdot x_{r'}^{t'}$ with $y_{r,r'}^{t,t'}$ in (1), the following two constraints should be added: $y_{r,r'}^{t,t'} \le x_r^t, y_{r,r'}^{t,t'} \le x_{r'}^{t'}, \forall\langle t,t'\rangle \in \mathcal{T}, \langle r,r'\rangle \in \mathcal{R}$. Thus, we can re-formalize (1) as follows:

$$\min \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}} x_r^t \cdot \alpha_r^t + \sum_{r,r'\in\mathcal{R}} y_{r,r'}^{t,t'} \cdot \beta_{r,r'} \cdot b_{t,t'}$$
$$S.t. \begin{cases} \sum_{r\in\mathcal{R}} x_r^t = 1, & \forall t\in\mathcal{T} \\ x_r^t \le I_r^t, & \forall t\in\mathcal{T}, r\in\mathcal{R} \\ x_r^t\cdot\tau_r \le \delta_r^t, & \forall t\in\mathcal{T}, r\in\mathcal{R} \\ y_{r,r'}^{t,t'}\cdot\tau_{r,r'} \le \delta_{t,t'}, & \forall t,t'\in\mathcal{T}, r,r'\in\mathcal{R} \\ \sum_{t,t'\in\mathcal{R}} y_{r,r'}^{t,t'}\cdot b_{t,t'} \le \mu_{r,r'}, & \forall r,r'\in\mathcal{R} \\ \sum_{t\in\mathcal{T}} x_r^t\cdot\varphi_t \le h_r, & \forall r\in\mathcal{R} \\ y_{r,r'}^{t,t'} \le x_r^t, & \forall t,t'\in\mathcal{T}, r,r'\in\mathcal{R} \\ y_{r,r'}^{t,t'} \le x_{r'}^{t'}, & \forall t,t'\in\mathcal{T}, r,r'\in\mathcal{R} \\ x_r^t\in\{0,1\} & \forall t\in\mathcal{T}, r\in\mathcal{R} \\ y_{r,r'}^{t,t'}\in\{0,1\} & \forall t,t'\in\mathcal{T}, r,r'\in\mathcal{R} \end{cases} \tag{3}$$

We can see that (3) is a 0-1 linear programming. However, it is still challenging to solve it even with a classical solution like the randomized rounding algorithm [40]. The main reason is that the dependency between variables $\{x_r^t\}$ and $\{y_{r,r'}^{t,t'}\}$ poses challenges to the rounding process. Specifically, if rounding $y_{r,r'}^{t,t'}$ first and then setting x_r^t is to 1 when $y_{r,r'}^{t,t'}$ is rounded to 1, we cannot guarantee that the access delay constraint and the communication delay constraints are strictly satisfied. Another rounding method is to round x_r^t first and then set $y_{r,r'}^{t,t'}$ to 1 when x_r^t is rounded to 1. However, this rounding method not only may violate the access delay and communication delay constraints, but also cannot guarantee the approximate performance of zone resource constraints. To break the dependency between variables $\{x_r^t\}$ and $\{y_{r,r'}^{t,t'}\}$, we give the following two definitions.

Definition 1: Given a task $t\in\mathcal{T}$, a subset $\mathcal{R}_t$ of zone set $\mathcal{R}$ is defined as available zone set for task t if any zone in $\mathcal{R}_t$ satisfies the service type, access delay and data locality constraints, *i.e.*, $x_r^t \le I_r^t$ and $x_r^t\cdot\tau_r^t \le \delta_r^t, \forall r\in\mathcal{R}_t$. If a tenant requires local data storage, R_t will only include cloud nodes within the tenant-specified geographic area.

Definition 2: Given a task pair of t and t', a zone pair of r and r' is defined as a feasible zone pair if it satisfies the communication delay constraint, *i.e.*, $\tau_{r,r'} \le \delta_{t,t'}$.

The feasible zone pair set of task pair $\langle t,t'\rangle$ is denoted as $A(t,t')$. With the help of Definition 1, the first and second sets of constraints in (3) can be dismissed. Besides, using Definition 2, the third set of constraints can also be deleted. Moreover, the cost of task placement can be rewritten as:

$$\sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\left(y_{r,r'}^{t,t'}\cdot\alpha_r^t + y_{r,r'}^{t,t'}\cdot\alpha_{r'}^{t'}\right) \tag{4}$$

Let $\theta(r,r')$ indicate whether r and r' represent the same zone or not. If r and r' are the same zone, $\theta(r,r')=1$; otherwise, $\theta(r,r')=0$. The resource consumption of task pair $\langle t,t'\rangle$ in zone r can be formalized as:

$$\sum_{r'\in R}\left[y_{r,r'}^{t,t'}\cdot\theta(r,r')\cdot(\varphi_t+\varphi_{t'}) + y_{r,r'}^{t,t'}\cdot\theta(r,r')\cdot\varphi_t\right] \tag{5}$$

Based on the above analysis, we can convert (3) into the following 0-1 linear programming:

$$\min \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{R}}\left(y_{r,r'}^{t,t'}\cdot\alpha_r^t + y_{r,r'}^{t,t'}\cdot\alpha_{r'}^{t'}\right) + \sum_{r,r'\in\mathcal{R}} y_{r,r'}^{t,t'}\cdot\beta_{r,r'}\cdot b_{t,t'}$$
$$S.t. \begin{cases} \sum_{r\in\mathcal{R}_t}\sum_{r'\in\mathcal{R}_{t'}} y_{r,r'}^{t,t'} = 1, & \forall\langle t,t'\rangle\in\mathcal{T} \\ \sum_{t,t'\in\mathcal{R}} y_{r,r'}^{t,t'}\cdot b_{t,t'} \le \mu_{r,r'}, & \forall(r,r')\in\mathcal{R} \\ \sum_{t\in\mathcal{T}}\sum_{t'\in\mathcal{T}_t}\sum_{r'\in\mathcal{R}} & \\ \Big[y_{r,r'}^{t,t'}\cdot\theta(r,r')\cdot(\varphi_t+\varphi_{t'}) & \\ \quad +y_{r,r'}^{t,t'}\cdot\theta(r,r')\cdot\varphi_t\Big] \le h_r, & r\in R \\ y_{r,r'}^{t,t'}\in\{0,1\} & \forall t\in\mathcal{T}, r\in\mathcal{R} \end{cases} \tag{6}$$

Note that there is only variables $y_{r,r'}^{t,t'}$ in (6), thus the dependency between variables $\{x_r^t\}$ and $\{y_{r,r'}^{t,t'}\}$ has been eliminated.

B. Algorithm to Solve CAP

Given these insights, we propose an efficient rounding-based task placement algorithm called RB-CAP for the CAP problem. Due to the difficulty of 0-1 integer program, in the first step, we replace $y_{r,r'}^{t,t'}\in\{0,1\}$ with $y_{r,r'}^{t,t'}\in[0,1]$ in (6), thereby constructing a linear programming. Let $\{\widetilde{y_{r,r'}^{t,t'}}\}$ be the optimal solution of the relaxed linear programming of (6). Since $\widetilde{y_{r,r'}^{t,t'}}$ is fractional, each task may be divided and placed in multiple zones, which is not feasible for the CAP problem. Thus, in the second step, we round off the fractional solution $\{\widetilde{y_{r,r'}^{t,t'}}\}$ to integer solution (denoted as $\{\hat{y}_{r,r'}^{t,t'}\}$). For each task pair $\langle t,t'\rangle$, $t,t'\in\mathcal{T}$ and feasible zone pair $\langle r,r'\rangle\in A(t,t')$, we set $\hat{y}_{r,r'}^{t,t'}=1$ with probability of $\widetilde{y_{r,r'}^{t,t'}}$. When $\hat{y}_{r,r'}^{t,t'}=1$, it means that tasks t and

Algorithm 1: RB-CAP: Rounding-based Task Placement Algorithm for CAP.

```
1:  Step 1: Solving the Relaxed CAP Problem
2:  Construct a linear programming by replacing the
    integral constraints $y^{t,t'}_{r,r'} \in \{0,1\}$ with $y^{t,t'}_{r,r'} \in [0,1]$ in
    (6)
3:  Obtain the optimal solutions $\{\widetilde{y^{t,t'}_{r,r'}}\}$
4:  Step 2: Select Zones to Place Tasks
5:  Derive an integer solution $\hat{y}^{t,t'}_{r,r'}$ by randomized
    rounding
6:  for each task pair $\langle t,t'\rangle, t,t' \in \mathcal{T}$ do
7:    for each feasible zone pair $(r,r') \in A(t,t')$ do
8:      Set $\hat{y}^{t,t'}_{r,r'} = 1$ with probability of $\widetilde{y^{t,t'}_{r,r'}}$
9:      if $\hat{y}^{t,t'}_{r,r'} = 1$ then
10:       Place tasks $t$ and $t'$ in zones $r$ and $r'$ respectively
11: Return $\{\widetilde{y^{t,t'}_{r,r'}}\}$ and $\{\hat{y}^{t,t'}_{r,r'}\}$
```

t' will be placed in zones r and r' respectively. We summarize RB-CAP in Algorithm 1.

C. Approximation Performance Analysis

This section analyzes the approximation performance of the RB-CAP algorithm. With Definitions 1 and 2, the service type constraint, access delay constraint and communication delay constraint are satisfied definitely. Thus, in this section, we only give the approximate performance for inter-zone bandwidth constraint, zone resource constraint and the optimization objective. We first give the following two famous theorems for probability analysis.

Theorem 1 (Union Bound [41]): Given a countable set of n events: $A_1, A_2, \ldots, A_n$, each event A_i happens with possibility $\mathbf{Pr}(A_i)$. Then, $\mathbf{Pr}(A_1 \cup A_2 \cup \ldots \cup A_n) \le \sum_{i=1}^{n} \mathbf{Pr}(A_i)$.

Theorem 2 (Chernoff Bound [42]): Given n independent variables: $y_1, y_2, \ldots, y_n$, where $\forall y_i \in [0,1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^{n} y_i]$. Then, $\mathbf{Pr}[\sum_{i=1}^{n} y_i \ge (1+\epsilon)\mu] \le e^{\frac{-\epsilon^2 \mu}{2+\epsilon}}$, in which ϵ is an arbitrary value greater than 0.

Assuming that the minimum inter-zone bandwidth between zones is denoted by μ_{min} and the minimum resource of a zone is defined by h_{min}. We define a constant value α as follows:

$$\lambda = \min\left\{\frac{\mu_{min}}{b_{t,t'}}, \frac{h_{min}}{2\cdot\varphi_t}, t,t' \in \mathcal{T}\right\} \tag{7}$$

In most practical scenarios, the traffic intensity of a task is usually much less than the inter-zone bandwidth and the resource consumption of a task is usually much less than a zone's resource capacity. Thus, we can reasonably assume that $\lambda \gg 1$.

Theorem 3: The proposed RB-CAP algorithm can achieve the approximation factor of $\frac{3\cdot\log h}{\alpha} + 3$ for the inter-zone bandwidth constraint.

Proof: In the first step, the RB-CAP algorithm derives the fractional solution $\{\widetilde{y^{t,t'}_{r,r'}}\}$ for the relaxed CAP problem. When $\widetilde{y^{t,t'}_{r,r'}}$ is rounded to 1, task t and t' will be placed in zone r and r', respectively. Thus, the expected bandwidth load of zone pair (r,r') from the data transfer between task t and t' can be defined as a random variable $\kappa^{t,t'}_{r,r'}$ as follows:

$$\kappa^{t,t'}_{r,r'} = \begin{cases} b_{t,t'}, & \text{with probability of } \widetilde{y^{t,t'}_{r,r'}} \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

It is obvious that random variables $\{\kappa^{t,t'}_{r,r'}\}$ are mutually independent according to the definition. The expected bandwidth load of zone pair (r,r') can be expressed as:

$$\mathbb{E}\left[\sum_{t,t'\in\mathcal{T}} \kappa^{t,t'}_{r,r'}\right] = \sum_{t,t'\in\mathcal{T}} \mathbb{E}[\kappa^{t,t'}_{r,r'}] = \sum_{t,t'\in\mathcal{T}} \widetilde{y^{t,t'}_{r,r'}} \cdot b_{t,t'} \le \mu_{r,r'} \tag{9}$$

(21) shows that the RB-CAP algorithm can guarantee that the expectation of bandwidth load of zone pair (r,r') is less than its capacity $\mu_{r,r'}$. Now we analyze the degree and probability of inter-zone bandwidth capacity constraints being violated. We combine the definition of λ in (7) and (21). It follows:

$$\begin{cases} \frac{\kappa^{t,t'}_{r,r'}\cdot\lambda}{\mu_{r,r'}} \in [0,1] \\ \mathbb{E}\left[\sum_{t,t'\in\mathcal{T}} \frac{\kappa^{t,t'}_{r,r'}\cdot\lambda}{\mu_{r,r'}}\right] \le \lambda \end{cases} \tag{10}$$

By applying Theorem 2 and assuming that ρ is an arbitrary positive value. We have:

$$\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}} \frac{\kappa^{t,t'}_{r,r'}\cdot\lambda}{\mu_{r,r'}} \ge (1+\rho)\cdot\lambda\right] \le e^{\frac{-\rho^2\cdot\lambda}{2+\rho}} \tag{11}$$

Now, we assume that:

$$\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}} \frac{\kappa^{t,t'}_{r,r'}}{\mu_{r,r'}} \ge (1+\rho)\right] \le e^{\frac{-\rho^2\cdot\lambda}{2+\rho}} \le \frac{1}{g^4} \tag{12}$$

where g is the number of cloud zones. The solution for (12) can be expressed as:

$$\rho \ge \frac{4\cdot\log g + \sqrt{16\cdot\log^2 g + 32\cdot\lambda\cdot\log g}}{2\lambda}$$
$$\Rightarrow \rho \ge \frac{4\log g}{\lambda} + 2, \quad g \ge 2 \tag{13}$$

Then using Union Bound Theorem, we have:

$$\mathbf{Pr}\left[\bigvee_{r,r'\in\mathcal{R}} \sum_{t,t'\in\mathcal{T}} \frac{\kappa^{t,t'}_{r,r'}}{\mu_{r,r'}} \ge (1+\rho)\right]$$
$$\le \sum_{r,r'\in\mathcal{R}} \mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}} \frac{\kappa^{t,t'}_{r,r'}}{\mu_{r,r'}} \ge (1+\rho)\right]$$
$$\le g^2 \cdot \frac{1}{g^4} = \frac{1}{g^2}, \quad \rho \ge \frac{4\log g}{\lambda} + 2 \tag{14}$$

The last inequality holds because there are at most g^2 inter-zone links given g zones. Eq (14) means that the proposed RB-CAP algorithm can guarantee that the total bandwidth load on any zone pair (r,r') will not exceed the fractional solution by a factor of $1+\rho = \frac{4\cdot\log g}{\lambda} + 3$ with high probability. ■

Theorem 4: The proposed RB-CAP algorithm can achieve the approximation factor of $\frac{3\cdot\log g}{\alpha}+3$ for the zone resource constraint.

Proof: The proof process is similar to that of Theorem 3. We first compute the expected resource consumption of task pair $\langle t,t'\rangle$, then present the degree and probability of zone resource constraints being violated. Let $\omega_r^{t,t'}$ be the expected resource consumption of task pair $\langle t,t'\rangle$ in zone r. We have:

$$\omega_r^{t,t'} = \sum_{r'\in R}\left[\widetilde{y_{r,r'}^{t,t'}}\cdot\theta(r,r')\cdot(\varphi_t+\varphi_{t'}) \quad +\widetilde{y_{r,r'}^{t,t'}}\cdot\theta(r,r')\cdot\varphi_t\right] \tag{15}$$

According to the definition of $\omega_r^{t,t'}$, they are independent of each other. The expected resource consumption of zone r can be expressed as:

$$\begin{aligned}\mathbb{E}\left[\sum_{t,t'\in\mathcal{T}}\omega_r^{t,t'}\right] &= \sum_{t,t'\in\mathcal{T}}\mathbb{E}[\omega_r^{t,t'}]\\ &= \sum_{t,t'\in\mathcal{T}}\sum_{r'\in R}\left[\widetilde{y_{r,r'}^{t,t'}}\cdot\theta(r,r')\cdot(\varphi_t+\varphi_{t'})+\widetilde{y_{r,r'}^{t,t'}}\cdot\theta(r,r')\cdot\varphi_t\right]\\ &\le h_r\end{aligned} \tag{16}$$

From (16), we can conclude that the expected resource consumption of zone r will not violate the zone resource constraint. Now we calculate the probability of actual resource consumption violating the zone resource constraints. Combining the definition of λ in (7) and (16), then applying Theorem 2, it follows:

$$\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\frac{\omega_r^{t,t'}\cdot\lambda}{h_r}\ge(1+\epsilon)\cdot\lambda\right]\le e^{\frac{-\epsilon^2\cdot\lambda}{2+\epsilon}} \tag{17}$$

where ϵ is an arbitrary positive value. Assuming that:

$$\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\frac{\omega_r^{t,t'}}{h_r}\ge(1+\epsilon)\right]\le e^{\frac{-\epsilon^2\cdot\lambda}{2+\epsilon}}\le\frac{1}{g^3} \tag{18}$$

Similar to (13), the solution for (18) can be expressed as:

$$\begin{aligned}&\epsilon\ge\frac{3\cdot\log g+\sqrt{9\cdot\log^2 g+24\cdot\lambda\cdot\log g}}{2\lambda}\\ &\Rightarrow \epsilon\ge\frac{3\log g}{\lambda}+2,\quad g\ge 2\end{aligned} \tag{19}$$

Using Union Bound Theorem, it follows:

$$\begin{aligned}&\mathbf{Pr}\left[\bigvee_{r\in\mathcal{R}}\sum_{t,t'\in\mathcal{T}}\frac{\omega_r^{t,t'}}{h_r}\ge(1+\epsilon)\right]\\ &\le\sum_{r\in\mathcal{R}}\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\frac{\omega_r^{t,t'}}{h_r}\ge(1+\epsilon)\right]\\ &\le g\cdot\frac{1}{g^3}=\frac{1}{g^2},\quad \epsilon\ge\frac{3\log g}{\lambda}+2\end{aligned} \tag{20}$$

(20) indicates that the proposed RB-CAP algorithm can guarantee that the total resource consumption of any zone $r\in\mathcal{R}$ will not exceed the fractional solution by a factor of $1+\epsilon=\frac{3\cdot\log g}{\lambda}+3$ with high probability. ■

We use O_{ALG} to denote the objective value of (6) associated with Alg. 1. Let O_{LP} be the optimal objective value to the relaxed version of (6). We have the following theorem regarding the approximate ratio of the optimization objective.

Theorem 5: the proposed RB-CAP algorithm can achieve an approximation factor of $O(\log g)$ for the objective value, that is, $O_{ALG}\le O(\log g)\cdot O_{LP}$.

Proof: We first analyze the approximation performance of task placement cost. The expected task placement cost of task pair $\langle t,t'\rangle$ is defined as $\nu_{t,t'}=\widetilde{y_{r,r'}^{t,t'}}\cdot\alpha_r^t+\widetilde{y_{r,r'}^{t,t'}}\cdot\alpha_{r'}^{t'}$. The total expected task placement cost can be expressed as:

$$\begin{aligned}\mathbb{E}\left[\sum_{t,t'\in\mathcal{T}}\nu_{t,t'}\right]&=\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\nu_{t,t'}]\\ &=\sum_{t,t'\in\mathcal{T}}\left(\widetilde{y_{r,r'}^{t,t'}}\cdot\alpha_r^t+\widetilde{y_{r,r'}^{t,t'}}\cdot\alpha_{r'}^{t'}\right)\end{aligned} \tag{21}$$

We define a constant as follows:

$$\zeta_{max}=\max\{2\cdot\alpha_r^t,\forall r\in\mathcal{R},t\in\mathcal{T}\} \tag{22}$$

Let $\chi_1=\frac{\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\nu_{t,t'}]}{\zeta_{max}}$. Using Theorem 2, we have:

$$\begin{aligned}&\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\frac{\nu_{t,t'}}{\zeta_{max}}\ge(1+\rho)\cdot\chi_1\right]\le e^{\frac{-\rho^2\cdot\chi_1}{2+\rho}}\\ &\Rightarrow\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\nu_{t,t'}\ge(1+\rho)\cdot\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\nu_{t,t'}]\right]\le e^{\frac{-\rho^2\cdot\chi_1}{2+\rho}}\end{aligned} \tag{23}$$

With the similar proof process to that of task placement cost, the data transfer cost satisfies:

$$\mathbf{Pr}\left[\sum_{t,t'\in\mathcal{T}}\varepsilon_{t,t'}\ge(1+\rho)\cdot\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\varepsilon_{t,t'}]\right]\le e^{\frac{-\rho^2\cdot\chi_2}{2+\rho}} \tag{24}$$

where $\chi_2=\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\varepsilon_{t,t'}]/\zeta'_{max}$ and $\zeta'_{max}=\max\{\beta_{r,r'}\cdot b_{t,t'},\forall r,r'\in\mathcal{R},t,t'\in\mathcal{T}\}$. Similar to (19), we can derive the following inequations from (23) and (24).

$$\begin{cases}\sum_{t,t'\in\mathcal{T}}\nu_{t,t'}\le O(\log g)\cdot\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\nu_{t,t'}]\\ \sum_{t,t'\in\mathcal{T}}\varepsilon_{t,t'}\le O(\log g)\cdot\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\varepsilon_{t,t'}]\end{cases} \tag{25}$$

Now, we we can conclude that:

$$\frac{O_{ALG}}{O_{LP}}=\frac{\sum_{t,t'\in\mathcal{T}}\nu_{t,t'}+\sum_{t,t'\in\mathcal{T}}\varepsilon_{t,t'}}{\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\nu_{t,t'}]+\sum_{t,t'\in\mathcal{T}}\mathbb{E}[\varepsilon_{t,t'}]}\le O(\log g) \tag{26}$$

It means that the RB-CAP algorithm can achieve an approximation factor of $O(\log g)$ for the objective value. ■

D. Complete Algorithm to Solve CAP

Though the RB-CAP algorithm can obtain the bi-criteria approximation performance for the CAP problem, the zone resource and bandwidth capacity constraints can not be always satisfied due to the randomized rounding process. Thus, we present a complete algorithm, called CRB-CAP, which can strictly meet all constraints. There are four steps in the CRB-CAP algorithm. The first and second steps are same as those in

Algorithm 2: CRB-CAP: Complete RB-CAP Algorithm.

```
1:  Step 1: Same as that in RB-CAP
2:  Step 2: Same as that in RB-CAP
3:  Step 3: Removing Tasks which Violate Zone
    Resource and Bandwidth Capacity Constraints
4:  $\mathcal{T}' \leftarrow \emptyset$
5:  for each zone $r \in \mathcal{R}$
6:    Rank tasks in zone $r$ with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$
7:    while the resource constraint of zone $r$ is violated do
8:      Remove the task with the minimum $\widetilde{y_{r,r'}^{t,t'}}$ and add
        this task to $\mathcal{T}'$
9:  for each zone pair $(r, r')$ do
10:   Rank tasks which occupy the bandwidth resource
      between zones $r$ and $r'$ with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$
11:   while the bandwidth capacity between zone $r$ and $r'$
      is violated do
12:     Remove the task with the minimum $\widetilde{y_{r,r'}^{t,t'}}$ add this
        task to $\mathcal{T}'$
13: Step 4: Reselecting Zones which Satisfying Zone
    Resource and Bandwidth Capacity Constraints
14: for each task $t \in \mathcal{T}'$ do
15:   Rank zones with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$
16:   Place task $t$ in zone with the maximum $\widetilde{y_{r,r'}^{t,t'}}$, while
      the zone resource and bandwidth capacity
      constraints are satisfied
```

RB-CAP. In the third step, CRB-CAP removes tasks that violate zone resource and bandwidth capacity constraints. The set of removed tasks is denoted as $\mathcal{T}'$. For each zone $r \in \mathcal{R}$, CRB-CAP ranks tasks in zone r with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$ and removes the task with the minimum $\widetilde{y_{r,r'}^{t,t'}}$ and add this task to $\mathcal{T}'$ until the resource constraint of zone r is satisfied. For each zone pair (r, r'), CRB-CAP ranks tasks that occupy the bandwidth resource between zones r and r' with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$. Then it removes the task with the minimum $\widetilde{y_{r,r'}^{t,t'}}$ and adds this task to $\mathcal{T}'$ until the bandwidth capacity between zone r and r' is satisfied. In the fourth step, CRB-CAP reselects zones with zone resource and bandwidth capacity constraints. Similarly, for each task $t \in \mathcal{T}'$, it ranks zones with increasing order of $\widetilde{y_{r,r'}^{t,t'}}$ and places task t in zone with the maximum $\widetilde{y_{r,r'}^{t,t'}}$, while the zone resource and bandwidth capacity constraints are satisfied. The CRB-CAP algorithm is summarized in Alg. 2.

E. Discussion

Limitations of CADER: 1) Lack of Support for Dynamic Task Scaling. The current model focuses on static task deployment optimization and does not yet cover dynamic scaling needs in burst traffic scenarios (*e.g.*, a temporary doubling of task instances during e-commerce promotional peaks). For tenants requiring such functionality, coordination between CADER and external elastic scheduling tools (*e.g.*, Kubernetes Horizontal Pod Autoscaler) is necessary. CADER is responsible for initial cross-cloud zone deployment planning of tasks; Kubernetes dynamically adjusts the number of task instances within the zones specified by CADER based on real-time traffic monitoring data, avoiding latency fluctuations caused by cross-zone scaling. 2) Cost Difference Compared to SkyPilot. Since CADER prioritizes meeting tenants' access latency constraints (*e.g.*, deploying tasks in geographically closer zones with slightly higher service prices), its deployment cost is approximately 6% higher on average than SkyPilot (which ignores latency constraints). A constraint priority setting can be added to CADER's tenant interface. If tenants prioritize cost, they can lower the latency constraint weight (*e.g.*., allowing a 5% exceedance of the latency threshold), and the system will prioritize low-cost zones during optimization. If tenants require strict low-latency guarantees (*e.g.*, real-time video transmission tasks), they can increase the latency constraint weight, in which case the system will automatically sacrifice partial cost advantages to meet latency requirements. The parameter range for this configuration (*e.g.*, adjustable upper/lower bounds of latency thresholds, estimated cost savings range) is displayed in real time on the interface, enabling tenants to make independent choices based on business needs.

Fault Handling: CADER focuses on optimization during the resource provisioning phase. For fault handling, two strategies are implemented: (1) Priority retry in the same zone strategy. When a task fails to execute due to non-node faults (such as temporary resource contention or software exceptions), it will first be retried in the original deployment zone. This strategy leverages the delay and traffic constraints already satisfied by the original zone, avoiding cost increases and delay fluctuations caused by cross-zone redeployment. (2) Node fault degradation strategy. If a zone-level fault (*e.g.*, node downtime, network outage) is detected, the system triggers a lightweight reselection process of the CRB-CAP algorithm: It filters the top-k feasible zones with the highest probabilities based on the original fractional solution $\widetilde{y_{r,r'}^{t,t'}}$, and prioritizes selecting nodes with the optimal bandwidth connectivity to the original zone for redeployment, ensuring task continuity.

Future research directions: (1) Deadline-constrained application deployment algorithm design. Our current work focuses on optimizing multi-cloud deployments primarily based on tenant requirements and resource constraints. However, many real-world applications, such as time-sensitive data processing, financial trading, and emergency response systems, impose strict deadlines on task completion. In future research, we plan to design a deployment algorithm that integrates completion time constraints into the optimization model. (2) Integrating cloud region carbon footprint into placement decisions. In alignment with the global push towards sustainable computing, a future research direction is to incorporate the carbon footprint of different cloud regions into deployment decision-making process.

V. Performance Evaluation

In this section, we perform comprehensive simulations to assess the efficacy of CADER. The simulation results indicate that:

1) CADER can reduce the overall cost by 16% –38% compared with DelayFirst and CostFirst, respectively, and the

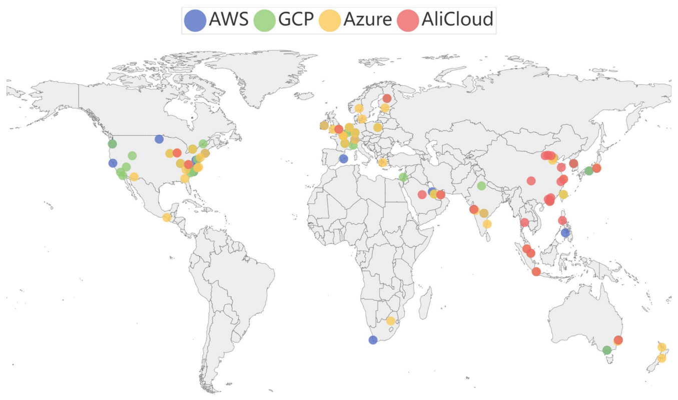


Fig. 4. Multi-cloud topology. Cloud zones from AWS, GCP, Azure and AliCloud are shown in the picture.

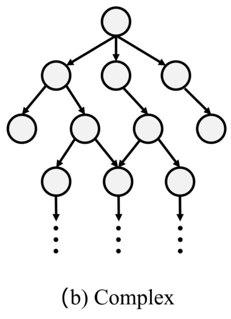


Fig. 5. Real DAGs found in Airflow's respository [48].

cost gap between CADER and SkyPilot is within 6% (Exp #1).

2) Given different average access delay demand and traffic demand, CADER achieves about 11% -37% reduction in deployment cost compared with DelayFirst and CostFirst (Exp #2 - Exp #4).

3) Compared with SkyPilot, CADER can reduce access delay and communication delay by about 31% and 40% , respectively (Exp #5 - Exp #6).

A. Simulation Settings

Cloud Settings: To accurately simulate a multi-cloud environment, we collect the cloud zone information from AWS [12], GCP [14], Azure [13] and AliCloud [43]. According to Synergy report [44], these four cloud service providers rank among the top global players with the highest cloud market share. We plot all the cloud zones of these four cloud service providers in Fig. 4. Users whose locations are randomly generated from various cities on land can access these zones. To establish the access delay between users and cloud zones, we employ the Euclidean distance method [45], mapping it to a range of 10 ms to 100 ms. Similarly, the delay between any two zones is also generated using the Euclidean distance method. The service prices are generated based on the official pricing information [12], [13], [14], [43]. Specifically, the price for 8 vCPUs with 16 GB of memory is set within the range of $0.289 \sim 0.49$\$per hour. The price for TPU v2/v3 is set within the range of $1.24 \sim 2.2$\$per hour, while the price for GPUs is set within the range of $0.355 \sim 0.487$\$per hour. The data transfer cost between zones from different cloud service providers is set at 0.8 \$/GB, whereas the data transfer cost between zones from the same provider ranges from $0.2 \sim 0.5$ \$/GB. Additionally, the default bandwidth between zones is set to 50 Gbps [46].

Cloud Application Settings: We utilize a real-world task dataset obtained from the Google Cluster Trace [47] to conduct simulations. However, this dataset does not provide information about the DAG of cloud applications. To address this limitation, we employ sequential DAGs and complex DAGs, which are commonly found in Airflow's repository [48]. These two types of DAGs, as depicted in Fig. 5, are representative of real-world scenarios [49]. In both types of DAGs, the resource types required for each task are randomly selected from CPU, GPU, and TPU. Furthermore, the resource requirements for each task are generated based on the Google Cluster Trace [47]. The data transmission rate between each pair of nodes is set to 10 GB per hour. In terms of the cloud application requirements, we assume that each task follows a uniform distribution with an average access delay of 50 ms. Similarly, the bandwidth and communication delay demands are randomly sampled from a uniform distribution, with an average delay of 50 ms and an average bandwidth demand of 50 Mbps. While we leverage simulation to evaluate the performance of CADER, we recognize that real-world validation is essential. In the future, we plan to conduct real-world multi-cloud deployments in actual multi-cloud environments. 1) Short-term: leverage official APIs of leading cloud service providers (AWS EC2 API, Azure Resource Manager API, GCP Compute Engine API, and AliCloud ECS API) to construct a small-scale multi-cloud testbed. This testbed will replicate the cloud zone topology, resource pricing, and network latency settings used in simulations. 2) Mid-term: deploy real-world workloads, such as distributed machine learning model training in real-world multi-cloud environments and optimize the algorithm's adaptability to non-ideal networks. Runtime logs will be collected, including records of non-ideal network events (*e.g.*, temporary bandwidth drops, API response delays). This data will be used to optimize CADER's algorithm. 3) Long-term: release CADER's prototype as an open-source project, with core modules (Information Collector, Deployment Optimizer, and Fault Handling Module) packaged as reusable SDKs. Function iterations will be driven by community feedback—for example, adding custom constraint templates (*e.g.*, industry-specific compliance templates for healthcare or finance) based on user requests.

B. Benchmarks and Performance Metrics

Benchmarks: We compare CADER against the following three widely used benchmarks in practical scenarios. Note that, due to the limited availability of research on task deployment across multiple clouds, the first two benchmarks have been adapted from existing intra-cloud deployment methods. The third benchmark represents an existing approach for multi-cloud deployment.

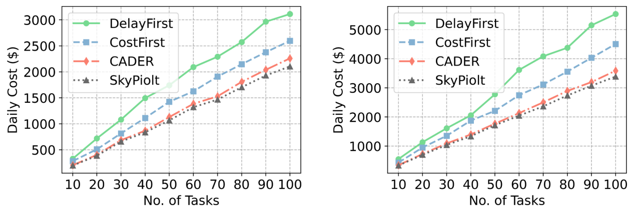
1) The first benchmark, DelayFirst, selects a zone with the minimum access delay for each task in the cloud application. This benchmark aims to simulate user behavior where lower access latency is desired to provide a better service experience [29].
2) The second benchmark, CostFirst, is a modification of the approach presented in [9]. For each task in the cloud application, CostFirst chooses a zone that satisfies the access delay constraint and has the lowest service price, but it disregards the diversity in transfer costs [9]. This benchmark helps verify that directly adopting existing inter-cloud task deployment methods may result in higher deployment costs, as it overlooks the significant differences in data transfer costs between zones from different cloud service providers.
3) The third benchmark, SkyPilot [4], applies an ILP solver to obtain the task deployment scheme with the minimum deployment cost. However, this method does not consider the access delay and traffic demand requirements. SkyPilot simulates user preferences for lower deployment costs. We use this benchmark to illustrate that, although CADER takes into consideration access delay and traffic demand requirements, its deployment cost remains comparable to that of SkyPilot.

Metrics: We evaluate the performance of CADER using the following three important metric. (1) Daily deployment cost. This metric represents the cost of deploying a cloud application in a single day. It consists of two components. The first component is the daily cost of service acquisition, which is calculated by multiplying the total resource consumption of tasks within the cloud application by the service price within a 24-hour period. The second component is the data transfer cost, which is computed by multiplying the total data transfer volume by the data transfer cost per unit. The sum of these two parts gives the daily deployment cost. (2) Access delay. This metric measures the delay experienced by users when accessing the purchased services. It is obtained by summing up the access delay of each task and dividing it by the total number of tasks. (3) Communication delay. This metric quantifies the delay between tasks. Similar to access delay, it is calculated by summing up the communication delay for all pairs of tasks involved in data transmission and dividing it by the total number of task pairs with data transmission. These last two metrics provide insights into the QoS experienced by users.

C. Evaluation Results

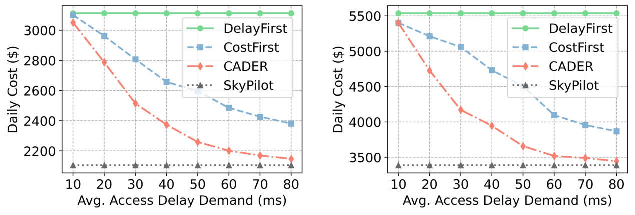
We first test the performance of CADER against other comparison algorithms in terms of daily deployment cost.

(Exp #1) Daily cost versus the number of tasks: In order to evaluate the performance of these algorithms in a more intuitive manner, we analyzed the daily cost by varying the number of tasks in a cloud application using two types of DAGs. The results are presented in Fig. 6. The graph shows that the daily cost of CADER is comparable to that of SkyPilot and significantly lower



(a) Sequential (b) Complex

Fig. 6. (Exp #1) Daily Cost vs. No. of Tasks.



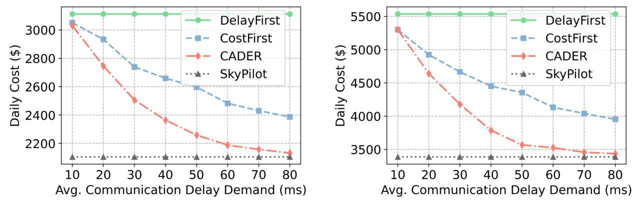
(a) Sequential (b) Complex

Fig. 7. (Exp #2) Daily Cost vs. Average Access Delay Demand.

than those of DelayFirst and CostFirst. Specifically, when considering 80 tasks with a sequential DAG, the daily costs of DelayFirst, CostFirst, CADER, and SkyPilot are 2573, 2146, 1807, and 1703, respectively. This means that compared to DelayFirst and CostFirst, CADER can achieve a reduction of 30% and 16% in the overall daily cost, respectively. The advantage of CADER lies in its ability to provide more advantageous deployment decisions from a global perspective, ensuring fair handling of tenant demands and zonal resource restrictions. Additionally, the daily cost gap between CADER and SkyPilot is approximately 6% . SkyPilot, while slightly outperforming CADER by about 6% , incurs a slightly higher daily cost due to its deployment decisions not considering access delay requirements and traffic demand, as shown in Fig. 6(b). For instance, when the number of tasks is 90, the daily costs of DelayFirst, CostFirst, CADER, and SkyPilot are \$5148, \$4030, \$3197, and \$3072, respectively. In this case, CADER can reduce the overall daily cost by 38% and 21% compared to DelayFirst and CostFirst, respectively, with a daily cost gap of approximately 4% compared to SkyPilot. We observe that CADER can achieve greater cost reduction than DelayFirst and CostFirst when dealing with complex task DAGs. This suggests that cloud applications with complex DAGs have a more significant impact on performance compared to those with sequential DAGs. Simple methods such as DelayFirst and CostFirst would result in much higher deployment costs.

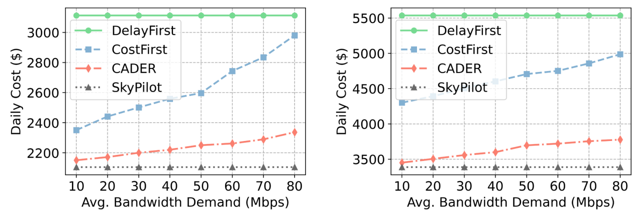
To showcase the high application of CADER across various scenarios, we conducted a comparative simulations given 100 tasks. We alter the tenant requirements, specifically focusing on access delay, communication delay and bandwidth demands. The corresponding outcomes are illustrated in Figs. 7- 9.

(Exp #2) Daily cost versus average access delay demand: Fig. 7 analyzes the influence of average access delay demand on the daily deployment cost. As the access delay demand increases, the daily costs of CostFirst and CADER decrease, while those of DelayFirst and SkyPilot remain unchanged. This can be attributed to DelayFirst's tendency to choose the closest



(a) Sequential (b) Complex

Fig. 8. (Exp #3) Daily Cost vs. Average Communication Delay Demand.

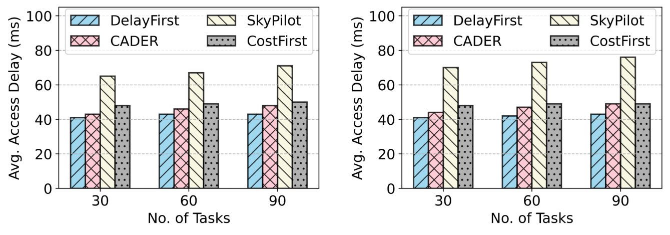


(a) Sequential (b) Complex

Fig. 9. (Exp #4) Daily Cost vs. Average Bandwidth Demand.

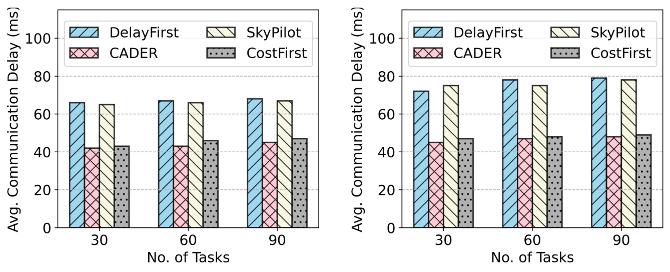
cloud zone for each task, which has the minimum access delay. On the other hand, SkyPilot does not consider the tenant's access delay demand, resulting in a constant daily cost regardless of the access delay demand. We observe that the daily cost of CADER is significantly lower than that of CostFirst and DelayFirst. For instance, when the average access delay demand is set to 60 ms under the sequential DAG in Fig. 7(a), the daily costs of CADER, DelayFirst, and CostFirst are \$3112, \$2485, and \$2201, respectively. Specifically, CADER achieves a cost reduction of approximately 29% and 11% compared to DelayFirst and CostFirst, respectively. This difference becomes even more pronounced when considering the complex DAG in Fig. 7(b), where the daily costs of CADER, DelayFirst, and CostFirst are \$5535, \$3996, and \$3518, respectively. In this case, CADER achieves a cost reduction of approximately 36% and 12% compared to DelayFirst and CostFirst, respectively.

(Exp #3) Daily cost versus average communication delay demand: The impact of average communication delay demand on the daily deployment cost is also evaluated in Fig. 8. The graph demonstrates the changes in daily cost as the average communication delay demand among tasks increases from 10 ms to 80 ms. It is observed that as the communication delay demand increases, the daily costs of CostFirst and CADER decrease, while those of DelayFirst and SkyPilot remain unchanged. This behavior can be attributed to the fact that DelayFirst and SkyPilot do not consider the communication delay demand, resulting in a constant daily cost regardless of the communication delay demand. Similar to the findings in Fig. 7, the daily cost of CADER is significantly lower than that of CostFirst and DelayFirst. For instance, considering an average communication delay demand of 70 ms under the sequential DAG in Fig. 8(b), the daily costs of CADER, DelayFirst, and CostFirst are \$5535, \$4040, and \$3456, respectively. This indicates that CADER can achieve a cost reduction of approximately 37% and 14% compared to DelayFirst and CostFirst, respectively. *(Exp #4) Daily costs versus average bandwidth demand:* In this series of experiments,



(a) Sequential (b) Complex

Fig. 10. (Exp #5) Average Access Delay vs. No. of Tasks.



(a) Sequential (b) Complex

Fig. 11. (Exp #6) Average Communication Delay vs. No. of Tasks.

we examine how the average bandwidth demand between tasks affects the daily deployment costs. The findings are depicted in Fig. 9, with the x-axis representing the average bandwidth demand between tasks, varying between 10 Mbps and 80 Mbps. The daily costs of CADER and CostFirst increase as the average bandwidth demand increases, while the daily costs of DelayFirst and SkyPilot remain constant. Notably, CADER consistently exhibits a much lower daily cost compared to CostFirst. For instance, when the average bandwidth demand is 60 Mbps in Fig. 9(a), the daily costs of DelayFirst, CostFirst, CADER, and SkyPilot are \$3112, \$2743, \$2261, and \$2104, respectively. Specifically, CADER achieves an average hourly cost reduction of approximately 27% and 18% compared to DelayFirst and CostFirst, respectively. This can be attributed to the fact that CostFirst only focuses on minimizing the service acquisition cost, while neglecting the data transfer cost. Furthermore, the daily cost difference between CADER and SkyPilot is approximately 7% . This discrepancy arises from the fact that SkyPilot does not consider the bandwidth requirement imposed by the tenant, making it easier to achieve a lower daily cost.

In order to further investigate the performance of these algorithms, we also analyze the average access delay of tasks and the average communication delay of task pairs for different numbers of tasks, ranging from 30 to 90. The results are depicted in Figs. 10 and 11.

(Exp #5) Access delay versus the number of tasks: In Fig. 10, we analyze the average access delay performance of the algorithms as the number of tasks increases. It is observed that DelayFirst achieves the lowest average access delay, while SkyPilot exhibits the highest access delay. The average access delays of CADER and CostFirst are relatively close to each other. For instance, considering a scenario with 60 tasks and a sequential DAG, as shown in Fig. 10(a), the average access delays of

DelayFirst, CADER, SkyPilot, and CostFirst are 43 ms, 46 ms, 67 ms, and 49 ms, respectively. This indicates that CADER can reduce the average access delay by approximately 31% compared to SkyPilot. The reason behind this improvement is that CADER typically chooses the cloud zone with the lowest access delay for each task. On the other hand, SkyPilot has the highest access delay because it solely focuses on minimizing the deployment cost, overlooking the access delay requirement.

(Exp #6) Communication delay versus the number of tasks: We observed the average communication delay performance of these algorithms for a range of 30 to 90 tasks in this set of experiments. The results show that DelayFirst and SkyPilot exhibit similar average communication delay performance, while CADER and CostFirst also demonstrate similar communication delay performance. Specifically, as shown in Fig. 10(b), when considering 60 tasks with a complex DAG, the average communication delays for DelayFirst, CADER, SkyPilot, and CostFirst are 78 ms, 47 ms, 75 ms, and 48 ms, respectively. This indicates that CADER reduces the average communication delay by approximately 40% and 37% compared to DelayFirst and SkyPilot, respectively. This improvement can be attributed to the fact that both CADER and CostFirst take into account the communication requirement when making deployment decisions, while SkyPilot and DelayFirst do not.

VI. Conclusion

In this paper, we study the problem of minimizing the deployment cost of cloud application with tenant requirement guarantee. We explore the challenges faced by the cloud application deployment problem in multi-clouds. To solve this problem, we design an efficient algorithm based on the randomized rounding method and prove its approximation factor. The efficacy of our algorithm is substantiated by in-depth analyses conducted on real-world datasets, showcasing its superiority over other solutions.

References

[1] "Disney streaming service," Accessed: Dec. 30, 2023. [Online]. Available: https://www.disneyplus.com/

[2] "Netflix streaming service," Accessed: Dec. 30, 2023. [Online]. Available: https://www.netflix.com/

[3] I. Stoica and S. Shenker, "From cloud computing to sky computing," in *Proc. Workshop Hot Topics Operating Syst.*, 2021, pp. 26–32.

[4] Z. Yang et al, "Skypilot: An intercloud broker for sky computing," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implementation*, 2023, pp. 437–455.

[5] "Multi-Cloud Strategy," 2023, Accessed: Dec. 30, 2023. [Online]. Available: https://www.digitalocean.com/resources/article/multi-cloud-strategy

[6] "Multicloud Infrastructure Provider Strategy," Accessed: Dec. 30, 2023. [Online]. Available: https://www.oracle.com/lu/news/announcement/98-percent-enterprises-adopted-multicloud-strategy-2023-02-09/

[7] "Best practices for choosing resource locations," 2021, Accessed: Dec. 30, 2023. [Online]. Available: https://cloud.google.com/solutions/best-practices-compute-engine-region-selection?hl=zh-cn

[8] R. Singh, S. Agarwal, M. Calder, and P. Bahl, "Cost-effective cloud edge traffic engineering with Cascara," in *Proc. 18th USENIX Symp. Netw. Syst. Des. Implementation*, 2021, pp. 201–216.

[9] L. Luo, G. Zhao, H. Xu, Z. Yu, and L. Xie, "Tango: A cost optimization framework for tenant task placement in geo-distributed clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2023, pp. 1–10.

[10] A. Isenko, R. Mayer, and H.-A. Jacobsen, "How can we train deep learning models across clouds and continents? an experimental study," 2023, *arXiv:2306.03163*.

[11] "TPU Cloud," 2025, Accessed: Dec. 30, 2023. [Online]. Available: https://github.com/google/cluster-data/

[12] "Amazon web services," Accessed: Dec. 30, 2023. [Online]. Available: https://aws.amazon.com/

[13] "Microsoft azure," Accessed: Dec. 30, 2023. [Online]. Available: https://azure.microsoft.com/en-us/

[14] "Google cloud," Accessed: Dec. 30, 2023. [Online]. Available: https://cloud.google.com/

[15] "Azure confidential computing," Accessed: Dec. 30, 2023. [Online]. Available: https://azure.microsoft.com/en-us/solutions/confidential-compute/

[16] "TPU Cloud," Accessed: Dec. 30, 2023. [Online]. Available: https://cloud.google.com/tpu

[17] "AWS Inferentia," Accessed: Dec. 30, 2023. [Online]. Available: https://aws.amazon.com/machine-learning/inferentia/

[18] "A clear multicloud strategy delivers business value," 2018, Accessed: Dec. 30, 2023. [Online]. Available: https://www.forrester.com/report/a-clear-multicloud-strategy-delivers-business-value/RES128781

[19] P. Jain, S. Kumar, S. Wooders, S. G. Patil, J. E. Gonzalez, and I. Stoica, "Skyplane: Optimizing transfer cost and throughput using cloud-aware overlays," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implementation*, 2023, pp. 1375–1389.

[20] N. Khorasani, "Cloud Broker: A systematic mapping study," *IEEE Trans. Serv. Comput.*, vol. 17, no. 5, pp. 2989–3005, Sep./Oct. 2024.

[21] "RightScale," Accessed: Dec. 30, 2023. [Online]. Available: https://www.flexera.com/

[22] "Apache Libcloud," Accessed: Dec. 30, 2023. [Online]. Available: https://libcloud.apache.org/

[23] "Apache jclouds," Accessed: Dec. 30, 2023. [Online]. Available: https://jclouds.apache.org/

[24] "Terraform," Accessed: Dec. 30, 2023. [Online]. Available: https://www.terraform.io/

[25] H. Tu, G. Zhao, H. Xu, and X. Fang, "Tenant-grained request scheduling in software-defined cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4654–4671, Dec. 2022.

[26] Y. Huang, H. Xu, H. Gao, X. Ma, and W. Hussain, "SSUR: An approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 2, pp. 670–681, Jun. 2021.

[27] J. Gao, H. Wang, and H. Shen, "Smartly handling renewable energy instability in supporting a cloud datacenter," in *Proc. 2020 IEEE Int. Parallel Distrib. Process. Symp.*, 2020, pp. 769–778.

[28] M. Wang, B. Cheng, S. Wang, and J. Chen, "Availability-and traffic-aware placement of parallelized SFC in data center networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 182–194, Mar. 2021.

[29] Y. Yue, B. Cheng, X. Liu, M. Wang, B. Li, and J. Chen, "Resource optimization and delay guarantee virtual network function placement for mapping SFC requests in cloud networks," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1508–1523, Jun. 2021.

[30] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "Dynamic task placement for deadline-aware IoT applications in federated fog networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1469–1478, Jan. 2021.

[31] Z. A. Mann, "Decentralized application placement in fog computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3262–3273, Dec. 2022.

[32] H. Zhao, S. Cen, and Y. Zhu, "The space above the sky: Uniting global-scale ground station as a service for efficient orbital data processing," in *Proc. 2024 IEEE 32nd Int. Conf. Netw. Protoc.*, 2024, pp. 1–11.

[33] C. Z. Radulescu, M. Radulescu, and F. G. Filip, "Cloud provider selection a complex multicriteria problem," *Romanian J. Inf. Sci. Technol.*, vol. 24, pp. 337–352, 2021.

[34] B. Singh, R. Kaur, M. Woodside, and J. W. Chinneck, "Low-power multi-cloud deployment of large distributed service applications with response-time constraints," *J. Cloud Comput.*, vol. 12, no. 1, p. 1, 2023.

[35] A. Y. Aldailamy, A. Muhammed, R. Latip, N. A. W. A. Hamid, and W. Ismail, "Online dynamic replication and placement algorithms for cost optimization of online social networks in two-tier multi-cloud," *J. Netw. Comput. Appl.*, vol. 224, 2024, Art. no. 103827.

[36] T. Zhu, M. A. Kozuch, and M. Harchol-Balter, "WorkLoadCompactor: Reducing datacenter cost while providing tail latency SLO guarantees," in *Proc. Symp. Cloud Comput.*, 2017, pp. 598–610.

[37] W. Li, X. Zhou, K. Li, H. Qi, and D. Guo, "Trafficshaper: Shaping inter-datacenter traffic to reduce the transmission cost," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1193–1206, Jun. 2018.

[38] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas.*, 2009, pp. 202–208.

[39] C. Guo et al, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proc. 2015 ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 139–152.

[40] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.

[41] D. Hunter, "An upper bound for the probability of a union," *J. Appl. Probability*, vol. 13, no. 3, pp. 597–603, 1976.

[42] M. Hellman and J. Raviv, "Probability of error, equivocation, and the Chernoff bound," *IEEE Trans. Inf. Theory*, vol. 16, no. 4, pp. 368–372, Jul. 1970.

[43] "AliCloud," Accessed: Dec. 30, 2023. [Online]. Available: https://www.aliyun.com/

[44] "Cloud market gets its mojo back; AI helps push Q4 increase in cloud spending to new highs," Accessed: May 29, 2025. [Online]. Available: https://www.srgresearch.com/articles/cloud-market-gets-its-mojo-back-q4-increase-in-cloud-spending-reaches-new-highs/

[45] P.-E. Danielsson, "Euclidean distance mapping," *Comput. Graph. Image Process.*, vol. 14, no. 3, pp. 227–248, 1980.

[46] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, "Cost-minimizing bandwidth guarantee for inter-datacenter traffic," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 483–494, Apr.–Jun. 2019.

[47] "Google Cluster," 2012, Accessed: Dec. 30, 2023. [Online]. Available: https://github.com/google/cluster-data/

[48] "Apache Airflow," Accessed: Dec. 30, 2023. [Online]. Available: https://airflow.apache.org/

[49] A. Mahgoub, E. B. Yi, K. Shankar, S. Elnikety, S. Chaterji, and S. Bagchi, "Orion and the three rights: Sizing, bundling, and prewarming for serverless dags," in *Proc. 16th USENIX Symp. Operating Syst. Des. Implementation*, 2022, pp. 303–320.

Huaqing Tu (Member, IEEE) received the BS degree in computer science from Northeastern University, China, in 2018, and the PhD degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2023. During 2023 2025, she was a postdoctoral fellow with Zhejiang Laboratory, China. She is currently an assistant researcher with the School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China. Her research interests mainly include software-defined networks, programmable networks, and cloud computing. She has authored or coauthored more than ten articles in famous journals and conferences, including *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Parallel and Distributed Systems*, *Computer Networks*, and IEEE/ACM International Symposium on Quality of Service.

Ziqiang Hua received the BS degree from Nanjing Normal University, Nanjing, China, in 2018, and the PhD degree from Zhejiang University, Hangzhou, China, in 2023. He is currently an assistant Researcher with Zhejiang Laboratory. His research interests include cloud computing, network control, and resource orchestration.

Qianpiao Ma received the BS degree in computer science and the PhD degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2014 and 2022, respectively. He is currently a postdoctoral researcher with Purple Mountain Laboratories, China. His research interests include cloud computing, federated learning, mobile edge computing, and distributed machine learning.

Hanguang Luo received the BS degree in communications engineering and the MS degree in signal processing from the Guilin University of Electronic Technology, Guilin, China, in 2010 and 2013, respectively, and the PhD degree in communication and information system from the University of Electronic Science and Technology of China, Chengdu, China, in 2019. He is currently a associate research fellow with the Zhejiang Laboratory, Hangzhou, China. His research interests include network security, software-defined networks, and future networks.

Tao Zou received the BS degree and the PhD degree in communication and information systems from the National University of Defense Technology, China, in 1997 and 2004, respectively. He is currently a research fellow with the Zhejiang Laboratory, China. His research interests include software-defined networks, future network, and industrial internet.

Gongming Zhao (Member, IEEE) received the PhD degree in computer software and theory from the University of Science and Technology of China, Hefei. China, in 2020. He is currently an associate professor with the University of Science and Technology of China. His research interests include software-defined networks, cloud computing, and networking for AI.

Hongli Xu (Member, IEEE) received the BS degree in computer science and the PhD degree in computer software and theory from the University of Science and Technology of China (USTC), Hefei, China, in 2002 and 2007, respectively. He is currently a professor with the School of Computer Science and Technology, USTC. He has authored or coauthored more than 100 articles in famous journals and conferences, including *IEEE/ACM Transactions on Networking*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, International Conference on Computer Communications (INFOCOM), and International Conference on Network Protocols (ICNP). He has held more than 30 patents. His research interests include software defined networks, edge computing, and the Internet of Thing. He was recipient of the Outstanding Youth Science Foundation of NSFC in 2018, and the Best Paper Award or the Best Paper Candidate in several famous conferences.