Toward a Service Availability-Guaranteed Cloud Through VM Placement

Jiawei Liu, Gongming Zhao, *Member, IEEE*, Hongli Xu, *Member, IEEE*, Peng Yang, Baoqing Wang, and Chunming Qiao, *Fellow, IEEE*

***Abstract*— In a multi-tenant cloud, the cloud service provider (CSP) leases physical resources to tenants in the form of virtual machines (VMs) with an agreed service level agreement (SLA). As the most important indicator of SLA, we should guarantee the service availability of tenants when placing the VMs. However, previous works about VM placement mainly concentrate on optimizing the cloud resource utilization, but only a few works consider the service availability by measuring the hardware availability. In fact, abnormal tenants can make the corresponding service unavailable by launching network attacks. That is, both the hardware availability and the tenant uncertainty will affect the service availability of VMs on physical machines (PMs). Without considering this factor, the CSP may fail to meet the tenant's SLA requirements, leading to a reduction in revenue. To solve such a problem, this paper considers the service availability in terms of both the hardware availability and the tenant uncertainty, and studies the service availability-guaranteed VM placement in multi-tenant clouds (SAG-VMP) problem. This problem is very challenging since the service availability actually changes with the tenants served on the PM. To address this issue, we propose a two-phase approach: PM assignment and VM placement. The first phase determines the availability of each PM through a long-term tenant-PM mapping algorithm and the second phase places each VM on a PM that meets the service availability requirement based on a primal-dual online algorithm. Two algorithms with bounded approximation factors are proposed for these two phases, respectively. Both small-scale experiment results and large-scale simulation results show the superior performance of our proposed algorithms compared with other alternatives.**

***Index Terms*— Cloud computing, multi-tenant, SLA, service availability, approximation, VM placement.**

Manuscript received 24 September 2023; revised 17 March 2024; accepted 10 May 2024; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor N. Karamchandani. This work was supported in part by the National Science Foundation of China (NSFC) under Grant 62102392, Grant 62132019, and Grant 62372426, in part by the National Science Foundation of Jiangsu Province under Grant BK20210121, in part by the Fundamental Research Funds for Central Universities, in part by Hefei Municipal Natural Science Foundation under Grant 2022013, and in part by the Youth Innovation Promotion Association of Chinese Academy of Science under Grant 2023481. *(Corresponding author: Gongming Zhao.)*

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

Baoqing Wang is with the School of Software Engineering, University of Science and Technology of China, Hefei, Anhui 230027, China, and also with Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu 215123, China (e-mail: bbq@mail.ustc.edu.cn).

Chunming Qiao is with the Department of Computer Science and Engineering, The State University of New York, Buffalo, NY 14260 USA (e-mail: drqiao01@gmail.com).

Digital Object Identifier 10.1109/TNET.2024.3401758

I. INTRODUCTION

CLOUD computing can significantly reduce tenants' cost of maintaining the physical infrastructure through centralized management, and thus more and more tenants (*e.g.*, enterprises and individual users) are migrating their tasks to the cloud [1]. In order to provide computation resources to tenants, Cloud Service Providers (CSPs), *e.g.*, Amazon Web Services (AWS) [2] and Google Cloud [3], virtualize computing resources as Virtual Machines (VMs), which are then leased to tenants. Considering the various requirements of different tenants and heterogeneity of physical machines (PMs), one of the most important problems faced by CSPs is VM placement.

In a multi-tenant cloud, the CSP provides VMs to tenants and gets paid from tenants. Each tenant has a service level agreement (SLA) with its CSP and one of the most important SLA indicators is *service availability* [4]. Service availability is the percentage of uptime of a VM, which can be measured by dividing the service available time by the total service cycle time. In practice, according to the SLA (*e.g.*, SLA of AWS [5] or Google Cloud [6]), when the service availability of VMs provided by CSP to tenants cannot meet the SLA targets, CSP should provide compensation to tenants, which will significantly reduce the revenue of CSP. For example, in the Compute Service Level Agreement for AWS [5], if a tenant's service availability target is 99.99%, CSP is required to refund the tenant at least 10% of the service fee when the actual uptime percentage of the tenant's VMs falls below that target. In order to maximize the revenue of CSP, we must *ensure the service availability when placing tenants' VMs*, which is challenging.

Previous works of VM placement mainly concentrate on optimizing the cloud resources in different aspects, such as maximizing resource utilization or load balancing [7], [8], [9], reducing network traffic or improving network performance [10], [11], saving cost or energy [12], [13]. Regretfully, these works do not consider the service availability requirements of tenants, thus may lead dissatisfaction to tenants and reduce the revenue of the CSP. To bridge the gap, the studies on service availability of cloud computing has attracted more and more attention [14], [15], [16], [17]. However, those works simply consider the availability of hardware equipment, *i.e.*, hardware availability.

In fact, in a multi-tenant cloud, *the availability of a PM is related to not only its hardware availability, but also the tenants deployed on this PM*. Specifically, the abnormal tenants (*e.g.*, malicious tenant) are common [18] in a multi-tenant cloud. These abnormal tenants may launch a

wide spectrum of network attacks, including denial of service (DoS) and co-residency attacks [19]. The attacks launched by abnormal tenants will send a large amount of traffic, reduce the performance of PMs, and even paralyze PMs, which affect the availability of the PMs. More seriously, once the shared physical resources (*e.g.*, caches and network adapter) are maliciously occupied by abnormal tenants, it affects the VMs of other tenants on the same PM. In this case, the availability of the PM is much lower than its hardware availability. Therefore, when it comes to the availability of the PMs in a multi-tenant cloud, the uncertainty of tenants (*i.e.*, the availability impact from tenants) served on PMs should not be ignored, which has not been noticed in previous works.

In this paper, we consider the service availability in terms of both *the hardware availability and the tenant uncertainty*, and study the problem of service availability-guaranteed VM placement in multi-tenant clouds (SAG-VMP). Note that, it is far from trivial to introduce the tenant uncertainty into the availability calculation of PMs. This is because the availability of the PM is uncertain and will vary with the tenants it serves. To conquer the above challenges, we solve the SAG-VMP problem by taking a two-phase approach: *PM assignment* and *VM placement*. For the first subproblem, we quantify the impact of tenant uncertainty on the availability of a PM and determine the availability of each PM through a long-term tenant-PM mapping algorithm. For the second subproblem, we use a primal-dual online algorithm to place each VM on a PM that meets the service availability requirement. The main contributions of this paper are as follows:

1) We are the first work to quantify the impact of tenant uncertainty on service availability in a multi-tenant cloud. We formulate the problem of service availability-guaranteed VM placement in multi-tenant clouds (SAG-VMP) and solve the problem by taking a two-phase approach: PM assignment and VM placement.
2) For the PM assignment subproblem, we prove it is NP-hard. We propose a randomized rounding based algorithm, and prove the approximation factor is $O(\log n)$, where n is the number of PM nodes in a data center.
3) For the VM placement subproblem, we prove it is also NP-hard. We design an online algorithm based on the primal-dual method. Moreover, we analyze that the proposed algorithm can achieve the competitive ratio of $[1/(1+\rho), O(\log J + \log(1/\rho))]$, where $\rho \in (0,1)$ and J is a system dependent constant.
4) We conduct small-scale tests and large-scale simulations using real-world topologies and datasets to show that the proposed algorithms achieve superior performance compared with state-of-the-art solutions.

The rest of this paper is organized as follows. Section II presents the related works. Section III introduces the system model, the problem statement and the algorithm workflow. In Section IV, we propose a rounding-based offline algorithm for the PM assignment subproblem. Section V gives a primal-dual based online algorithm to solve the VM placement subproblem. The experiment and simulation results are given in Section VI. In Section VII, we conclude this paper.

II. Related Work

To achieve a service availability-guaranteed cloud, the CSP expects to meet the service availability of served tenants while obtaining the maximum revenue. In this section, we summarize the state-of-the-art VM placement solutions and studies on the service availability of cloud computing.

In recent years, a series of VM placement mechanisms have been widely proposed, focusing on optimizing cloud resources from various perspectives. Maximizing resource utilization and load balancing [7], [8], [9] in the cloud are among the most prominent research areas in VM placement. Chhabra and Singh [7] propose the Optimal VM Placement for load balancing mechanism, which utilizes maximum likelihood estimation for parallel and distributed applications. Their approach improves throughput and reduces failure rate by considering CPU, memory, and energy estimations. Kumar et al. [8] develop a novel load-balancing framework that aims to minimize the operational cost of data centers by improving resource utilization. Their framework employs a modified genetic algorithm to achieve an optimal allocation of VMs on physical machines. Yao et al. [9] address resource fragmentation, inefficient utilization, and energy wastage in cloud data centers through a load balancing strategy based on virtual machine consolidation. Their strategy includes load state classification, resource-weight based VM selection, and VM placement algorithms to optimize resource utilization. Reducing network traffic and improving network performance are also active research areas in VM placement [10], [11]. Farzai et al. [10] propose a combined meta-heuristic communication-aware VMP approach, which reduces the total traffic load on network links by placing high-affinity VMs in close proximity whenever possible, particularly designed for VL2 topology. Xing et al. [11] present a traffic-aware ant colony optimization algorithm for VMP, incorporating innovative schemes for PM selection, VM ordering, and information exchange. Saving cost and energy [12], [13] is another major research direction in VM placement. Abbasi-khazaei and Rezvani [12] propose a multi-objective VMP approach that addresses the challenge of balancing renewable and fossil energy consumption in cloud data centers, while considering the time-constraint nature of IoT requests. Banerjee et al. [13] tackle the challenge of reducing energy consumption and improving resource utilization efficiency in cloud data centers with a novel game-theoretic approach.

While previous works in VM placement mainly concentrate on optimizing resource utilization, the importance of service availability in cloud computing has gained increasing attention [14], [15], [16], [17]. Lin et al. [14] propose a failure prediction technique based on historical data to identify faulty nodes in cloud service systems, enabling better VM allocation and migration for improved availability. Yang et al. [15] propose a delay-sensitive and reliable VM placement strategy to improve application placement availability in the cloud. Similarly, Liu et al. [17] develop a multi-objective optimization approach that considers virtual cluster availability, energy consumption, resource utilization, and load balance to achieve a trade-off among these objectives. Both studies aim to ensure service availability by assigning VMs with distinct service availability requirements to PMs with varying hardware availability levels. Zhao et al. [16] propose a multi-constraint VM

TABLE I
CLASSIFYING STATE-OF-THE-ART VM PLACEMENT SOLUTIONS IN CLOUD COMPUTING BASED ON VARIOUS TOPICS

Topic	Work	Key Approach	Focus
Resource Load	[7]	The maximum likelihood estimation	Improve throughput and reduce failure rate
	[8]	A modified genetic algorithm	Improve resource utilization, minimize operational cost
	[9]	The strategy based on VM consolidation	Address resource fragmentation, inefficient utilization
Network Traffic	[10]	Place high-affinity VMs nearby	Reduce the total traffic on network links
	[11]	A traffic-aware ant colony algorithm	Optimize PM selection, VM ordering, and traffic size
Cost & Energy	[12]	A multi-objective VMP approach	Balance renewable and fossil energy consumption
	[13]	The game-theoretic approach	Reduce energy consumption, improve efficiency
Service Availability	[14]	A failure prediction technique	Better VM migration to improve availability
	[15]	A delay-sensitive and reliable VM placement	Improve application placement availability
	[16]	A multi-constraint VM placement	Alleviate the impact of abnormal events
	[17]	A multi-objective optimization approach	Trade-off among availability, energy consumption, etc.

placement scheme to mitigate the impact on service availability caused by PM failures by limiting the number of tenants hosted by each PM. In conclusion, while previous works have acknowledged the significance of service availability in VM placement problems, they have primarily focused on hardware availability and overlooked the influence of tenant uncertainty on service availability. As far as we know, this is the first study to quantify the impact of tenant uncertainty on service availability within a multi-tenant cloud environment.

III. PRELIMINARIES

A. System Model

This section introduces the system model in multi-tenant clouds, including cloud infrastructure model, multi-tenant model and service availability model. For the ease of reference, the key notations in this paper are summarized in Table II.

1) Cloud Infrastructure Model: A typical cloud consists of several data centers and each data center consists of many PMs. For example, by the end of November 2021, Google Cloud has more than 200 data centers around the world [3], one of which consists of 10047 PMs [20]. In this paper, we focus on the VM placement problem in a data center. Let $P = \{p_1, p_2, \dots p_{|P|}\}$ denote the set of PM, where $|P|$ is the number of PMs in the data center. Since a PM is composed of many components (*e.g.*, CPU, memory and hard disk), we use a tuple (p, c) to represent the component c of the PM p. Let $R(p, c)$ denote the resource capacity of component c of PM p. Furthermore, we use R_p to denote the resource capacity of each PM $p \in P$, *i.e.*, R_p is the vector of $R(p, c)$.

2) Multi-Tenant Model: In a multi-tenant cloud, it usually consolidates VMs from different tenants atop a powerful PM. Let $T = \{t_1, t_2, \dots t_{|T|}\}$ denote a set of $|T|$ tenants. For each tenant $t \in T$, we represent the set of requested VMs as $V_t = \{v_t^1, v_t^2, \dots v_t^{|V_t|}\}$, where $|V_t|$ is the number of VMs required by tenant t. Each VM v of tenant t will consume some resources, such as CPU and RAM, denoted as D_t^v. Then we use $D_t = \sum_{v \in V_t} D_t^v$ to denote the resource demand of tenant t. Similar to R_p, both D_t^v and D_t are also vectors. Let Q_t^v denote the revenue of CSP from VM v of tenant t. Then, the revenue of CSP from tenant t is denoted as $Q_t = \sum_{v \in V_t} Q_t^v$. Let $A_t \in [0, 1]$ represent the promised service availability of the CSP to tenant $t \in T$, which is known in advance by the CSP. Specifically, if the SLA target of service availability of a tenant t is 99.99%, we have $A_t = 0.9999$.

3) Service Availability Model: To meet the service availability of the served tenants, a natural method is to deploy the tenants' VMs on PMs that meet the availability requirements [15]. Similar to prior studies [15], [21], the hardware availability of a PM component can be measured by its mean time to failure (MTTF) and mean time to repair (MTTR). Let $A(p, c)$ denote the hardware availability of component (p, c), which can be calculated as follows:

$$A(p, c) = \frac{MTTF(p, c)}{MTTF(p, c) + MTTR(p, c)} \tag{1}$$

In reality, the MTTF and MTTR of each component can be obtained from its factory documents and detailed maintenance logs [14], [15], respectively. As discussed in works [15], [21], [22], [23], the hardware availability of a PM is determined by the availability of its components. For instance, if the CPU of a PM fails, the PM becomes unavailable. Similarly, abnormalities in memory can also affect the usability of the PM. Let's use $\mathcal{C}_p$ to denote the component set of PM p, we can

TABLE II
KEY NOTATIONS

Parameters	Description
P	a PM set
R_p	the resource capacity of PM $p \in P$
A_p	the hardware availability of PM $p \in P$
A'_p	the availability of PM $p \in P$
T	a tenant set
T_r	the served tenant set in the data center
P_t	the feasible PM set of tenant $t \in T_r$
V	a VM set
V_t	the VM set of tenant $t \in T$
D_t^v	the resource requirement of VM $v \in V_t$
D_t	the resource requirement of tenant $t \in T$
Q_t^v	the revenue of CSP from VM $v \in V_t$
Q_t	the revenue of CSP from tenant $t \in T$
A_t	the service availability of tenant $t \in T$
μ_t	the abnormal ratio of tenant $t \in T$

Variables	Description
z_t	whether tenant $t \in T$ can be served by the data center
y_t^p	whether the CSP chooses PM $p \in P$ to provide services for the tenant $t \in T$
x_t^p	the demand proportion of tenant $t \in T$ served by PM $p \in P$
$f_{t,v}^p$	whether the VM $v \in V_t$ of tenant $t \in T$ is placed on PM $p \in P$

derive the hardware availability of the PM p as follows:

$$A_p = \prod\nolimits_{c \in \mathcal{C}_p} A(p, c) \tag{2}$$

Obviously, $A_p \in [0, 1]$. In this paper, since our focus is not on how to calculate the hardware availability of a PM, we assume that hardware availability A_p of each PM p is known.

In addition, the availability of a PM is closely related to the served tenants. Let μ_t denote the probability that the tenant t is an abnormal tenant (*i.e.*, abnormal rate), which can be obtained through its historical behavior and credit level [24]. Similar to the hardware availability of a PM, since our focus is not on determining whether tenant t is an abnormal tenant or not. In this paper, we assume the abnormal rate μ_t of a tenant t is known. Let y_t^p represent whether the CSP chooses PM p to provide services for tenant t or not. The availability A'_p of the PM p can be calculated as follows:

$$A'_p = \max\left\{0, A_p - \sum\nolimits_{t \in T} y_t^p \cdot \mu_t\right\}, \ \forall p \in P \tag{3}$$

B. Problem Statement

Given the complexity of the SAG-VMP problem, this section provides an overview. For a detailed exposition, refer to Section IV and Section V, which aim to enhance the clarity and completeness of our paper. For each VM placement request from different tenants, the data center must determine whether to accommodate the VM and which PM it should be placed on. In general, the goal of SAG-VMP is to pursue a maximum revenue of CSP as in many previous works [25], [26]. To achieve a service availability-guaranteed cloud, there are two specific constraints when pursuing the maximum revenue.

1) *PM Resource Constraint:* For each PM p in the data center, the resources used to place VMs cannot exceed its resource capacity R_p.
2) *Service Availability Constraint:* To ensure the service availability of served tenants. On the one hand, the availability A'_p of each PM p should be considered from both the hardware availability and the tenant uncertainty, as discussed in Section III-A.3. On the other hand, each VM of the served tenants should be placed on the PMs that meet its service availability requirement.

Here the revenue is as a case study in a cloud environment. It is worth noting that our problem and the proposed algorithm can easily be transformed into throughput maximization or other problems and corresponding algorithms.

C. Algorithm Workflow

In Section III-B, we summarize two critical constraints in the SAG-VMP problem, *i.e.*, PM resource constraint and service availability constraint. It should be noted that the availability of a PM is closely related to the tenants it serves, as discussed in Section III-A.3. In other words, when placing VMs, the availability of PMs is uncertain and changing with the tenants it serves. Thus, the SAG-VMP problem actually contains two sub-problems: *PM assignment* and *VM placement*. Specifically, the first subproblem determines the availability of each PM through a long-term tenant-PM mapping and the second subproblem places each VM on a PM that meets the service availability requirement.

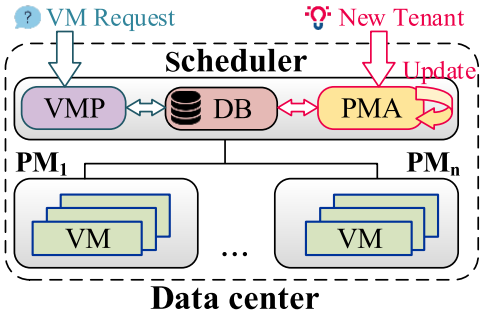

Fig. 1. Algorithm Workflow.

One may think that it is natural to design an algorithm to jointly solve the above two sub-problems. However, since these two sub-problems have some inherent differences, it may not be feasible. Specifically, if we update the tenant-PM mapping (*i.e.*, PM assignment) in the data center, the availability of many PMs will be changed and the service availability of a large number of VMs will be affected. To ensure the service availability of tenants, after each tenant-PM mapping update, the CSP needs to migrate VMs that do not meet the service availability requirements to the new PMs. It will significantly increase control overhead of CSP to maintain the consistency of VMs before and after migration. Thus, *we should reassign PMs for tenants in a long-term interval*. Meanwhile, tenants may create/delete VMs dynamically [1] and the faster the CSP responds to a tenant's VM placement request, the higher degree of satisfactory the tenants will experience [27]. Thus, *it is necessary to place VMs in an online manner*.

To this end, as depicted in Fig. 1, this paper solves the SAG-VMP problem through two phases: *PM assignment (PMA)* and *VM placement (VMP)*. Both PMA and VMP are executed on the schedulers within the data center. PMA is triggered when a new tenant joins the system and undergoes periodic updates over a longer time scale, such as daily. For this phase, we propose a rounding-based offline algorithm to determine the availability of each PM (Section IV). In contrast, VMP operates in real-time to efficiently manage and respond to VM requests from tenants. For this phase, we present a primal-dual based online algorithm to place each VM (Section V). Additionally, the information synchronization of two phases is achieved through a database (DB).

IV. PM Assignment

In this section, we first formulate the PM assignment (PMA) problem as a mixed integer programming problem. To solve this problem, we propose an approximation algorithm and analyze the algorithm performance.

A. Problem Definition for PMA

In order to determine the availability of each PM and meeting the service availability of each served tenant at a long-term interval, the PMA problem focuses on selecting which tenants to serve and choosing a feasible set of PMs for each served tenant. Specifically, given a set of PMs P and a set of tenants T in a data center, let $z_t \in \{0, 1\}$ indicate whether the tenant t is served by the data center or not. We use a set T_r to denote the served tenants, *i.e.*, $T_r = \{t | t \in T, z_t = 1\}$. Let binary variable y_t^p indicate whether the CSP chooses PM p to provide services for the tenant t or not. We use a set

P_t to denote the feasible set of PMs for each served tenant $t \in T_r$, *i.e.*, $P_t = \{p | p \in P, y_t^p = 1\}$. Let $x_t^p \in [0,1]$ denote the demand proportion of tenant t obtained from PM p. A feasible PM assignment scheme should satisfy the following constraints:

1) *Tenant Deployable Constraint:* For each served tenant t (*i.e.*, $z_t = 1$), we should provide all the services they need. That means, for each tenant $t \in T$, $\sum_{p \in P} x_t^p = z_t$.
2) *PM Resource Constraint:* The resources provided by PM p cannot exceed its resource capacity R_p. That means, for each PM $p \in P$, $\sum_{t \in T} x_t^p \cdot D_t \leq R_p$.
3) *PM Availability:* The availability of each PM requires to consider both the hardware availability and the tenant uncertainty. That means, for each PM $p \in P$, $A'_p = A_p - \sum_{t \in T} y_t^p \cdot \mu_t$.
4) *Service Availability Constraint:* Each tenant should satisfy its service availability at a long-term interval, *i.e.*, all VMs of tenant t should be placed on PM p that can meet its service availability requirement A_t. That means, for each tenant $t \in T$ and each PM $p \in P$, $A'_p \geq A_t \cdot y_t^p$.

We aim to find a feasible PM-tenant mapping solution with a maximum revenue of the CSP. Thus, the PMA problem can be formulated as follows:

$$\max \quad \sum_{t \in T} z_t \cdot Q_t$$

$$S.t. \begin{cases} x_t^p \leq y_t^p, & \forall t \in T, p \in P \\ y_t^p \leq z_t, & \forall t \in T, p \in P \\ \sum_{p \in P} x_t^p = z_t, & \forall t \in T \\ \sum_{t \in T} x_t^p \cdot D_t \leq R_p, & \forall p \in P \\ A'_p = A_p - \sum_{t \in T} y_t^p \cdot \mu_t, & \forall p \in P \\ A'_p \geq A_t \cdot y_t^p, & \forall t \in T, p \in P \\ x_t^p \in [0,1], & \forall t \in T, p \in P \\ y_t^p \in \{0,1\}, & \forall t \in T, p \in P \\ z_t \in \{0,1\}, & \forall t \in T \end{cases} \tag{4}$$

The first set of inequalities indicates whether some VMs of tenant t will be placed on PM p or not. The second set of inequalities indicates that only if the data center provides services to the tenant t, the VMs of tenant t can be placed on the PMs in this data center. The third set of equations represents the tenant deployable constraint. The fourth set of inequalities denotes the PM resource constraint. The fifth set of equations represents the PM availability. The last set of inequalities denotes the service availability constraint. Our objective is to maximize the revenue of CSP, *i.e.*, $\max \sum_{t \in T} z_t \cdot Q_t$.

Theorem 1: The PMA problem is NP-hard.

Proof: We consider a special example of the PMA problem, in which there is no constraint on the service availability (*i.e.*, $A_t = 0, \forall t \in T$), and assume that no tenant in the cloud is an abnormal tenant (*i.e.*, $\mu_t = 0, \forall t \in T$). Then, this becomes a Multi-dimensional 0-1 knapsack problem (MKP) [28], which is NP-Hard. Since the MKP problem is a special case of our problem, the PMA problem is NP-Hard too. □

B. Algorithm Design for PMA

Since the variables z_t and y_t^p are binary, it is difficult to solve PMA in a timely manner. Accordingly, in this section, we propose a rounding-based algorithm for the PMA problem, called RD-PMA. This algorithm consists of four steps. The first step relaxes Eq. (4) by replacing the eighth and ninth lines of integer constraints with $y_t^p \in [0,1]$ and $z_t \in [0,1]$, which reduces the PMA problem to be a linear programming. We can solve it with a linear programming solver (e.g., Cplex [29]) and obtain the optimal fractional solutions $\{\widetilde{x}_t^p\}, \{\widetilde{y}_t^p\}$, and $\{\widetilde{z}_t\}$. In the second step, we determine the tenants to serve, *i.e.*, obtain feasible solutions $\{\widehat{z}_t\}$. We first initialize an empty set T_r. For each individual tenant t, RD-PMA rounds variables $\{\widehat{z}_t\}$ to 1 with probability $\{\widetilde{z}_t\}$. Note that each rounding decision is independent of each other. If $\widehat{z}_t = 1$, CSP will use the data center to provide services to the tenant t, then we add tenant t into the served tenant set T_r. In the third step, we assign a feasible PM set to each served tenant, *i.e.*, obtain feasible solutions $\{\widehat{y}_t^p\}$ and $\{\widehat{x}_t^p\}$. For each tenant $t \in T_r$, we first initialize an empty set P_t. Then, for each PM $p \in P$, we set $\widehat{y}_t^p \leftarrow 1$ with probability $\widetilde{y}_t^p / \widetilde{z}_t$. If $\widehat{y}_t^p = 1$, we add the chosen PM p into feasible PM set P_t and set $\widehat{x}_t^p \leftarrow \widetilde{x}_t^p / \widetilde{y}_t^p$. In the fourth step, we update the availability of each PM. For each PM $p \in P$, we calculate its availability by $A'_p \leftarrow A_p - \sum_{t \in T_r} \widehat{y}_t^p \cdot \mu_t$. After the above four steps, we can obtain the served tenant set T_r, the feasible PM set P_t of each tenant $t \in T_r$ and the availability A'_p of each PM $p \in P$. The RD-PMA algorithm is formally described in Algorithm 1.

Algorithm 1 RD-PMA: Rounding-based Algorithm for PMA

```
Input: System Model in Section III-A
Step 1: Sloving the relaxed PMA problem
Construct a linear program by replacing with y_t^p ∈ [0,1]
  and z_t ∈ [0,1] in Eq. (4)
Derive the optimal fractional solutions {x̃_t^p, ỹ_t^p, z̃_t}
Step 2: Selecting tenants to serve
Initialize the served tenant set T_r = ∅
for each tenant t ∈ T do
    Initialize ẑ_t = 0
    Set ẑ_t ← 1 with probability z̃_t
    if ẑ_t == 1 then
        Set T_r ← T_r ∪ {t}
Step 3: PM assignment for served tenants
for each tenant t ∈ T_r do
    Initialize the feasible PM set P_t = ∅
    for each PM p ∈ P do
        Initialize x̂_t^p = 0 and ŷ_t^p = 0
        Set ŷ_t^p ← 1 with probability ỹ_t^p / z̃_t
        if ŷ_t^p == 1 then
            Set x̂_t^p ← x̃_t^p / ỹ_t^p and P_t ← P_t ∪ {p}
Step 4: Updating PM availability
for each PM p ∈ P do
    A'_p ← A_p − Σ_{t∈T_r} ŷ_t^p · μ_t
Output: T_r, P_t1, P_t2, ... P_t|T_r|, A'_p1, A'_p2, ... A'_p|P|
```

C. Performance Analysis

Theorem 2: The Algorithm 1 can achieve the bi-criteria approximation of $(\log n/\gamma + 1, \log n/\tau + 1)$, where n denotes the number of PMs in the data center, $\gamma = \min\left\{R_p^{min}/D_t, t \in T\right\}$ and $\tau = \min\{A_p^{min}/\mu_t, t \in T\}$. It means that the RD-PMA algorithm can achieve the optimal solution, violating the PM capacity by at most a factor

$\frac{\log n}{\gamma} + 1$, violating the service availability of tenant by at most a factor $\frac{\log n}{\tau} + 1$.

Proof: We analyze the approximate ratio based on the randomized rounding method [16], [30]. The detailed proof has been relegated to Appendix B-A. □

Theorem 3: The time complexity of the Algorithm 1 is $O((2|T||P| + |T|)^{2+1/6} \log(2|T||P| + |T|))$, where $|T|$ represents the number of tenants, and $|P|$ represents the number of PMs in the data center.

Proof: The time complexity of the RD-PMA algorithm can be divided into the following four parts. In Step 1, we use a linear programming solver (*e.g.*, Cplex [29]) to obtain the optimal fractional solution. According to state-of-the-art work [31], linear programming problems can be solved with a time complexity of $O(n^{2+1/6} \log(n/\delta))$, where δ represents the relative accuracy, and n denotes the number of variables. In the context of the PMA problem, the variables include x_t^p, y_t^p, and z_t. Hence, we have $n = 2|T||P| + |T|$ variables, where $|T|$ represents the number of tenants, and $|P|$ represents the number of PMs in the data center. By setting $\delta = 1$, the time complexity of Step 1 becomes $O((2|T||P| + |T|)^{2+1/6} \log(2|T||P| + |T|))$. In Step 2, we iterate over the tenants to determine which tenants to serve, the time complexity is $O(|T|)$. In Step 3, we first initialize an empty feasible PM set P_t for each tenant t. Then, we iterate over each PM p to determine whether to insert it into P_t. Hence, the time complexity of Step 3 is $O(|T||P|)$. In Step 4, we iterate over each PM to determine its availability A'_p, the time complexity is $O(|P|)$. In conclusion, the time complexity of our proposed RD-PMA algorithm is $O((2|T||P|+|T|)^{2+1/6} \log(2|T||P|+|T|))+O(|T|)+O(|T||P|)+O(|P|) = O((2|T||P|+|T|)^{2+1/6} \log(2|T||P|+|T|))$. □

V. VM Placement

In this section, we formalize the VM placement (VMP) problem as an integer linear programming problem. To solve this problem, we propose an online algorithm and analyze its competitive ratio.

A. Problem Definition for VMP

By the RD-PMA algorithm as described in Section IV, on the one hand, we determine which tenants provide services, *i.e.*, derive the feasible tenant set T_r. On the other hand, we achieve PM-tenant mapping and derive the feasible PM set P_t for each tenant $t \in T_r$. It is worth noting that VMs placed on the set P_t by the tenant $t \in T_r$ can satisfy the service availability requirements. Therefore, when discussing online VM placement, our objective is to maximize the revenue of CSP, and constraint is the PM resource constraint. Accordingly, let $f_{t,v}^p$ denote whether VM v of tenant t is placed on PM p or not. We formulate the VMP problem as follows:

$$\max \sum_{p\in P_t} \sum_{v\in V_t} \sum_{t\in T_r} f_{t,v}^p \cdot Q_t^v$$

$$S.t. \begin{cases} \sum_{p\in P_t} f_{t,v}^p \le 1, & \forall t \in T_r, v \in V_t \\ \sum_{t\in T_r} \sum_{v\in V_t} f_{t,v}^p \cdot D_t^v \le R_p, & \forall p \in P \\ f_{t,v}^p \in \{0,1\}, & \forall t \in T_r, v \in V_t, p \in P_t \end{cases} \tag{5}$$

The first inequality indicates that each VM v of tenant t can only be placed on one PM $p \in P_t$ at most. The second set of inequalities describes the resource constraint on each PM. Our goal is to achieve the maximum revenue in the data center.

Theorem 4: The VMP problem is NP-hard.

Proof: Obviously, the MKP [28] problem is a special case of the VMP problem. Since the MKP problem is NP-hard, VMP is NP-hard too. □

B. Online Algorithm Design for VMP

To solve the problem in Eq. (5), we design a primal-dual online algorithm for VMP (PD-VMP). At first, similar to the linear relaxation of Eq. (4), we construct the linear relaxation of Eq. (5) by replacing the binary variables $f_{t,v}^p \in \{0,1\}$ with continuous variables $f_{t,v}^p \in [0,1]$. Then, we consider the dual problem for the linear relaxation problem. Let $\alpha(t,v)$ and $\beta(p)$ be the dual variables of the first and second set of inequalities, respectively All these dual variables are non-negative. The dual problem of VMP can be described as follows:

$$\min \sum_{v\in V_t} \sum_{t\in T_r} \alpha(t,v) + \sum_{p\in P} R_p \cdot \beta(p)$$

$$S.t. \begin{cases} \alpha(t,v) + \beta(p) \cdot D_t^v \ge Q_t^v, & \forall t \in T_r, v \in V_t, p \in P_t \\ \alpha(t,v) \ge 0, & \forall t \in T_r, v \in V_t \\ \beta(p) \ge 0, & \forall p \in P \end{cases} \tag{6}$$

We can rewrite the first constraint of Eq. (6) as:

$$\alpha(t,v) \ge Q_t^v \left(1 - \frac{D_t^v}{Q_t^v} \beta(p)\right), \ \forall t \in T_r, v \in V_t, p \in P_t \tag{7}$$

The first step of the PD-VMP is to initialize corresponding constants and all the dual variables. According to the first set of inequalities in Eq. (6), we define a constant J as follows:

$$J = \max \left\{ \frac{D_t^v}{Q_t^v}, t \in T, v \in V_t \right\} \tag{8}$$

In addition, let's $\varphi = J/\rho$, where $\rho \in (0,1)$ is a parameter provided by PD-VMP, which trades off the optimization objective of algorithm and the meeting of quality of service requirements. The second step of PD-VMP is to select an appropriate PM for each arrival VM placement request. Upon the arrival of a new request, the algorithm calculates the price of each candidate PM, which is defined as:

$$K_p = \frac{D_t^v}{Q_t^v} \cdot \beta(p), \ \forall t \in T_r, v \in V_t, p \in P_t \tag{9}$$

The PD-VMP algorithm figures out the PM with lowest price, denoted as K_{p^*}. If $K_{p^*} \ge 1$, the constraints of the dual program will be violated (see proof of Lemma 5). Then, the request will be rejected and the corresponding dual variable $\alpha(t,v)$ will be set as 0. Otherwise, the task will be accepted and the algorithm will update the dual variables as follows:

$$\begin{cases} \alpha(t,v) \leftarrow Q_t^v (1 - K_{p^*}) \\ \beta(p^*) \leftarrow \beta(p^*) \left(1 + \frac{D_t^v}{R_{p^*}}\right) + \frac{D_t^v}{\varphi \cdot R_{p^*}} \end{cases} \tag{10}$$

The PD-VMP algorithm is formally outlined in Algorithm 2.

Algorithm 2 PD-VMP: Primal-Dual Online Algorithm for VMP

1: **Step 1: Algorithm initialization**
2: $\varphi = J/\rho$, where $\rho \in (0,1)$
3: Initialize the dual variables:
$\alpha(t,v) = 0, \forall t \in T_r, v \in V_t; \beta(p) = 0, \forall p \in P$
4: **Step 2: On arriving of a VM** $v \in V_t$ **request from tenant** $t \in T_r$
5: **for** each VM v placement request from tenant t **do**
6: **for** each $p \in P_t$ **do**
7: Calculate the price $K_p = \frac{D_t^v}{Q_t^v} \cdot \beta(p)$
8: $p^* \leftarrow \arg\min_{p \in P_t} K_p$
9: **if** $K_{p^*} < 1$ **then**
10: Place the VM v of tenant t on PM p^*
11: Set $\alpha(t,v) \leftarrow Q_t^v (1 - K_{p^*})$
12: Set $\beta(p^*) \leftarrow \beta(p^*) \left(1 + \frac{D_t^v}{R_{p^*}}\right) + \frac{D_t^v}{\varphi \cdot R_{p^*}}$.
13: **else**
14: Reject the VM v placement request

C. Performance Analysis of PD-VMP

We first prove the feasibility of the proposed PD-VMP algorithm.

Theorem 5: When the PD-VMP algorithm terminates, the primal-dual algorithm will not violate the constraints in the dual program Eq. (6).

Proof: There are two constraints in Eq. (6). On the one hand, we consider the second set of constraints in Eq. (6). Initially, all the dual variables are 0, which satisfies the positivity constraint. When dual variables $\beta(p)$ are updated, they will never decrease according to the update rules in Eq. (10). Besides, the update rule for dual variables, namely $\alpha(t,v) \leftarrow Q_t^v (1 - K_{p^*})$, can guarantee that $\alpha(t,v)$ is positive because only the VM placement request with $K_{p^*} < 1$ will be accepted. Therefore, the second set of inequalities in Eq. (6) is satisfied after PD-VMP terminates.

On the other hand, let's consider the constraints from the first set of inequalities in Eq. (6). For each VM placement request, we will determine a suitable PM p^* with the lowest price K_{p^*}. If it is rejected, according to Line 9 of Algorithm 2, $K_{p^*} \geq 1$ and the right side of Eq. (7) will not be positive. Since the dual variable $\alpha(t,v)$ is nonnegative, the first set of constraints in Eq. (6) is satisfied. If it is accepted, according to the update rule of Algorithm 2 and the definition of K_p of PM p, it follows:

$$\alpha(t,v) = Q_t^v (1 - K_p) \geq Q_t^v \left(1 - \frac{D_t^v}{Q_t^v} \cdot \beta(p)\right) \tag{11}$$

The above inequality is the same as the first set of constraints in Eq. (6). Moreover, the update of dual variables $\beta(p)$ will only make the right side smaller, which will not violate the constraints. As a result, the first set of constraints in Eq. (6) is always satisfied as well. □

To evaluate the performance of the proposed algorithm, we define the competitive ratio as follows [16] and [32].

Definition 1: If an online algorithm achieves at least $\zeta \cdot OPT$, where OPT is the result of the optimal solution, and the constraints are violated at most by a multiplicative factor η, we call that the online algorithm is $[\zeta, \eta]$ competitive.

Obviously, we expect the performance of PD-VMP is close to that of the optimal solution, *i.e.*, $\zeta \rightarrow 1$, and $\eta \rightarrow 1$. However, since the PD-VMP problem is NP-hard, it is infeasible to reach the above expectation for any polynomial time algorithm.

Theorem 6: The proposed Algorithm 2 can achieve the competitive ratio of $[1/(1+\rho), O(\log J + \log(1/\rho))]$, where ρ is an arbitrary parameter with $\rho \in (0,1)$, and J is a system dependent constant.

Proof: We analyze the approximate ratio based on comparison with the optimal solution. The detailed proof has been relegated to Appendix B-B. □

Theorem 7: The time complexity of the Algorithm 2 is $O(|P_t|)$ for each VM placement request, where $|P_t|$ represents the number of PMs in the feasible PM set P_t of tenant t determined by the Algorithm 1.

Proof: The time complexity of the PD-VMP algorithm can be divided into two parts. In Step 1, we perform an initialization with a time complexity of $O(1)$. In Step 2, when a VM v request arrives from a tenant t, we iterate over the feasible PM set P_t of the tenant. For each candidate PM p in P_t, we calculate the price K_p and record the lowest price as K_{p^*}. Then, we determine the placement scheme and update the dual variables based on K_{p^*}. Therefore, the time complexity Step 2 is $O(|P_t|)$, where $|P_t|$ represents the number of PMs in P_t. In conclusion, for each VM placement request, the time complexity of the PD-VMP algorithm is $O(1) + O(|P_t|) = O(|P_t|)$. □

VI. Performance Evaluation

In this section, we first introduce the metrics and benchmarks for performance comparison. Then, we compare our algorithm with the state-of-the-art solutions through large-scale simulations. Finally, we present the experimental results of a small-scale testbed based on Openstack [33].

A. Performance Metrics and Benchmarks

1) Performance Metrics: We adopt the following three sets of metrics to evaluate our proposed algorithms.

1) The satisfaction of tenants will affect the revenue of the CSP. Thus, the first set of metrics includes *the number of dissatisfied days of served tenants*, *the raw revenue* and *the SLA revenue*. Specifically, we first measure the daily failure time of each tenant and calculate the actual uptime percentage. We next compare the actual uptime percentage with the SLA target of service availability to determine whether the tenant is satisfied with the service of the day. For example, if the actual uptime percentage of a served tenant's VMs is 99.9% and the SLA target is 99.99%, then the tenant is unsatisfied on that day. Secondly, we calculate the total price of the resources sold by the CSP when SLAs are not considered and record it as the raw revenue. In contrast, we calculate the actual revenue that the CSP ultimately receives from tenants when considering compensation for services that fail to meet SLA targets and record it as the SLA revenue, *i.e.*, the SLA revenue is equal to the raw revenue minus the service compensation. We refer to Amazon Compute Service Level Agreement [5] to

TABLE III

AMAZON COMPUTE-BASED SERVICE LEVEL AGREEMENT [5]. IF THE SLA TARGET OF A TENANT'S SERVICE AVAILABILITY IS 99.9%, CSP NEEDS TO REFUND 10% OF THE SERVICE FEE TO THE TENANT WHEN THE ACTUAL UPTIME PERCENTAGE OF THE TENANT'S VMS IS LESS THAN 99.9% BUT GREATER THAN 99.8%

SLA Target	Uptime percentage	Compensation
99.9%	99.8% - 99.9%	10%
	99.5% - 99.8%	30%
	Less than 99.5%	100%
99.99%	99.98% - 99.99%	10%
	99.95% - 99.98%	30%
	Less than 99.95%	100%

calculate the specific service compensation, as shown in Table III.

2) The second set of metrics includes *the abnormal rate of served tenants*, *the unavailability of PMs* and *the number of affected PMs*. Specifically, when many tenants request services, the CSP may not be able to provide services for all tenants. Thus, we obtain the abnormal rate μ_t of the served tenants and calculate its average value. As discussed in Section III-A.3, the availability A'_p of a PM p can be calculated by $A'_p = A_p - \sum_{t \in T} y_t^p \cdot \mu_t$. Thus, we obtain the unavailability of a PM p by $1 - A'_p = 1 - A_p + \sum_{t \in T} y_t^p$. Additionally, when a tenant is an abnormal tenant, the attacks it launches may affect all the PMs serving it. Therefore, we measure the number of PMs that each tenant t maps to and record the value as the number of affected PMs, *i.e.*, $\sum_{p \in P} y_t^p$.

3) Abnormal tenants may affect the performance of the network. Thus, we adopt *the round-trip time (RTT)*, *the packet loss rate* and *the flow completion time (FCT)* as the third set of metrics. Specifically, we adopt iPerf3 [34] tool to measure the FCT of each served tenant, use ping tool to measure the RTT and packet loss rate of each PM.

2) Benchmarks: We divide SAG-VMP into two subproblems: PMA and VMP. Accordingly, we propose the RD-PMA algorithm for PMA and the PD-VMP algorithm for VMP. For simplicity, we denote the combined algorithm for SAG-VMP as **RD+PD**. We choose three benchmarks to compare with RD+PD. The first benchmark is the **NOVA** schedule scheme employed by OpenStack [35]. Upon receiving a VM placement request, a filtering algorithm is used to obtain a list of candidate PMs that possess the necessary resources and meet the VM requirements. Subsequently, a weighting algorithm is applied to rank the PMs and select the most suitable one. By default, the lower the load of the PM, the higher the ranking. The second benchmark is the rounding-based algorithm for VM placement that can alleviate the impact of abnormal events, called **R-VMP-AI** [16]. R-VMP-AI strategically limits the number of PMs allocated to each tenant and the number of tenants hosted per PM by setting thresholds derived from historical data (see detial in Section VI-B.1). This reduces the impact of malicious tenants and VM failures, improving service availability. However, R-VMP-AI lacks quantification of the impact of abnormal events on availability, and does not consider VM placement based on tenant SLA.

TABLE IV

MULTIPLE VM INSTANCES WITH DETAILED RESOURCES DEMAND AND PRICES FROM AMAZON EC2 CLOUD [36], [37]

Name	vCPU	RAM (GB)	Price ($ / h)
m5.large	2	8	0.096
m5.xlarge	4	16	0.192
m5.2xlarge	8	32	0.384
m5.4xlarge	16	64	0.768
c5.large	2	4	0.085
c5.xlarge	4	8	0.17
c5.2xlarge	8	16	0.34
c5.4xlarge	16	32	0.68
r5.large	2	16	0.126
r5.xlarge	4	32	0.252
r5.2xlarge	8	64	0.504
r5.4xlarge	16	128	1.008

The last benchmark is the delay-sensitive and reliable placement (**DSR**) algorithm [15]. To minimize the number of PMs used, DSR assigns PMs to VMs sequentially until all VMs are hosted without violating the VM availability constraint. DSR is a heuristic algorithm that uses the hardware availability of PMs to measure service availability but does not take into account the unavailability caused by tenant uncertainty.

B. Simulation Evaluation

1) Simulation Settings: Simulations are conducted on two practical topologies: the Koala-Based cloud [16] and the Google-Based cloud [20]. The Koala-Based cloud is a small-scale data center comprising 480 PMs. It focuses on individual tenants, with a default of 960 tenants who create 1-10 VM instances randomly. The Google-Based cloud, on the other hand, is a large-scale data center with 10,047 PMs. It mainly accommodates enterprise tenants, with the number of tenants set as 2,000 by default, and each tenant randomly creates 1-100 VM instances. In the benchmark, R-VMP-AI requires the CSP to set two appropriate thresholds: each PM can serve a maximum of h tenants, and each tenant's VM can be deployed on up to w PMs. Following the suggestion of Zhao et al. [16], the Koala-Based cloud sets h to 10 and w to 4, while the Google-Based cloud sets h to 10 and w to 40. To define the SLAs for the tenants, two different types are generated based on the Amazon Compute Service Level Agreement [5], as shown in Table III. For the VM instances, we generate three different types: standard (m5), computing-optimized (c5), and memory-optimized (r5), as shown in Table IV. These instance types are sourced from the Amazon EC2 cloud [36], [37], each with varying resource requirements (vCPU, RAM) and prices. For each tenant, their SLA and VM placement requests are randomly generated according to Tables III and IV, respectively. Notably, the probability of a tenant being abnormal follows a log-normal distribution, which effectively models real-world cyber attack magnitudes [38], [39]. A report [40] indicates that there are currently over 166,000 registered hackers worldwide, with an annual growth rate of approximately 10 percent. This suggests an average of one hacker for every 40,000 people in the population. Hence, the median of the distribution is set to 0.25 per ten thousand. In addition, for each PM node, we assume it has 64 CPU cores and 256GB RAM. Consistent

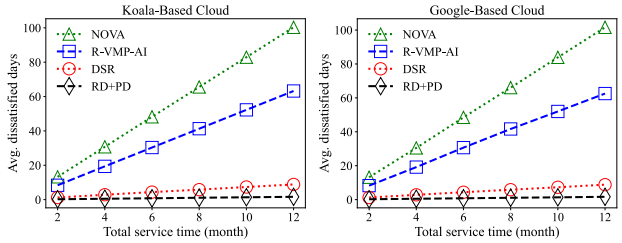

Fig. 2. Average number of dissatisfied days of served tenants vs. Total service time.

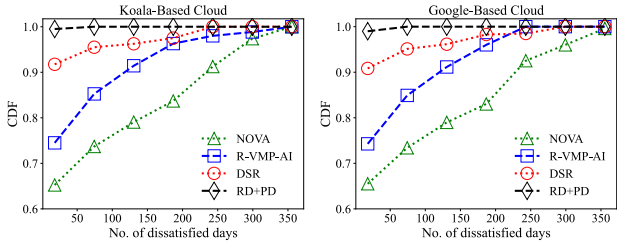

Fig. 3. CDF of the number of dissatisfied days of served tenants.

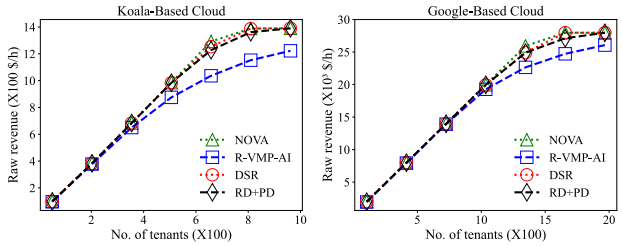

Fig. 4. Raw revenue of CSP vs. Number of tenants.

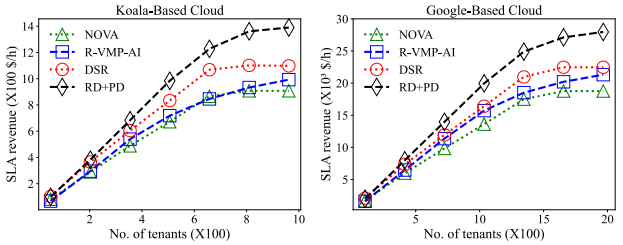

Fig. 5. SLA revenue of CSP vs. Number of tenants.

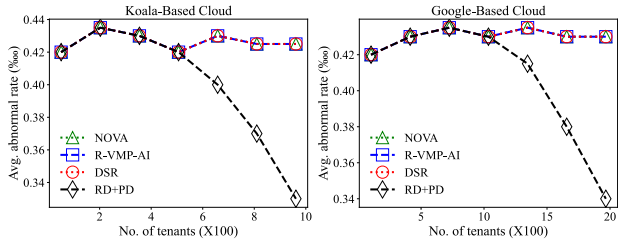

Fig. 6. Average abnormal rate of served tenants vs. Number of tenants.

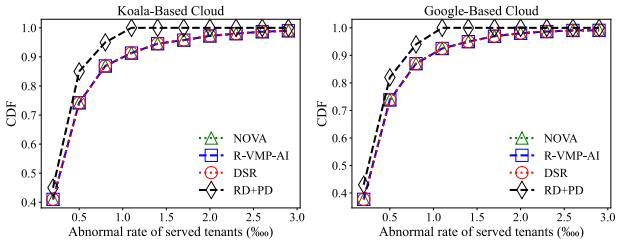

Fig. 7. CDF of the abnormal rate of served tenants.

with previous studies [41], [42], the hardware availability of each PM is randomly chosen from the set {99.95%, 99.99%, 99.995%, 99.999%}. We run each simulation for 30 trials, and the average results are shown.

2) Simulation Results: We run seven sets of simulations to evaluate the performance of the RD+PD algorithm against the other three benchmarks. The first set of simulations compares tenant satisfaction by changing the total service time under different algorithms and the results are shown in Figs. 2-3. We compare tenant satisfaction by calculating the average number of dissatisfied days of all served tenants in one year. Fig. 2 indicates that for all algorithms in both topologies, the number of dissatisfied days gradually increases with the number of days that tenants are served. We claim that RD+PD can reduce the number of dissatisfied days. For example, the average number of dissatisfied days per tenant in a year for our proposed algorithm is 0.9 days. In comparison, the average number of dissatisfied days are 10.3, 99.1 and 61.4 days for DSR, NOVA and R-VMP-AI, respectively. Fig. 3 shows the CDF of the number of dissatisfied days. We observe that the number of dissatisfied days of all tenants is less than 15 days by using the RD+PD algorithm, which is more than 200 by adopting other benchmarks. This remarkable improvement can be attributed to our proposed algorithm's comprehensive consideration of service availability, encompassing both hardware availability and tenant uncertainty.

The second set of simulations compares the revenue of CSP by changing the number of tenants in clouds. In Figs. 4-5, with the increasing number of tenants, the raw revenue and the SLA revenue of CSP increase for all algorithms in both topologies. Specifically, RD+PD achieves a similar performance in terms of the raw revenue compared to the other three benchmarks. In comparison, the proposed RD+PD algorithm can obtain a maximum SLA revenue of $1395/h in the Koala-Based cloud, while DSR, R-VMP-AI and NOVA can obtain SLA revenue of $1093/h, $985/h and $915/h, respectively. In other words, RD+PD can improve SLA revenue by about 27.6%, 41.6% and 52.4% compared with DSF, R-VMP-AI and NOVA, respectively. The reason is that we take into account the impact of tenant uncertainty on service availability and design an efficient algorithm to maximize the revenue of the CSP.

The third set of simulations compares the abnormal rate of served tenants by changing the number of tenants in clouds and the results are shown in Figs. 6-7. Fig. 6 shows that the average abnormal rate of our algorithm gradually decreases as the number of tenants increases, and is lower than other benchmarks. In comparison, the average abnormal rate of other benchmarks changes steadily. For example, when there are 960 tenants in the Koala-Based cloud, the average abnormal rate of served tenants is 0.33‰ by adopting RD+PD, while is about 0.43‰ by using other benchmarks. This is because when there are a large number of tenants, it is impossible to meet the VM placement requests of all tenants due to the resource constraint of the cloud. Our algorithm will first provide services to the tenants with good behavior in the history and high credit level (*i.e.*, tenants with lower abnormal rate), so the abnormal rate of the served tenants will gradually decrease with the increase of the number of tenants. Fig. 7 shows the CDF of the abnormal rate of served tenants. We observe that by adopting RD+PD, the abnormal rate of

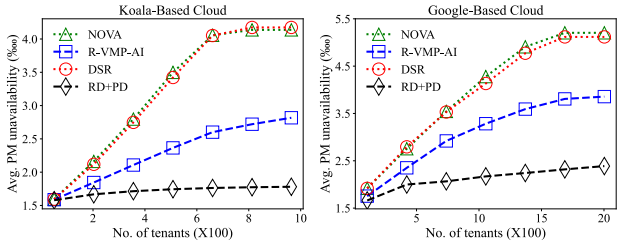

Fig. 8. Average unavailability of PMs vs. Number of tenants.

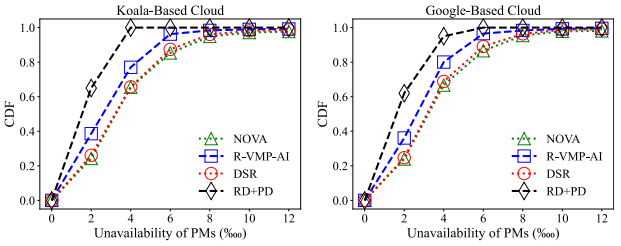

Fig. 9. CDF of the unavailability of PMs.

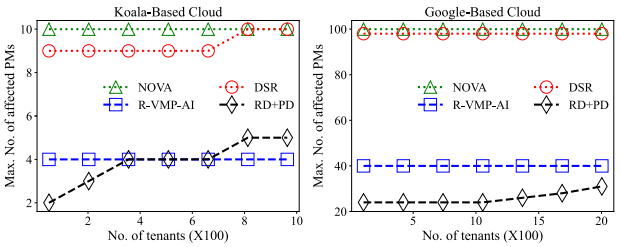

Fig. 10. The maximum number of PMs that a tenant can affect vs. Number of tenants.

all served tenants less than $1.5‰$ while the abnormal rate using other benchmarks can be greater than $3.0‰$.

The fourth set of simulations compares the unavailability of PMs by changing the number of tenants in clouds. As shown in Fig. 8, the unavailability of PMs for all algorithms in both topologies increases as the number of tenants increases. For instance, in the Koala-Based cloud with 960 tenants, the unavailability of PMs is $1.83‰$ when employing RD+PD. In comparison, DSR, NOVA, and R-VMP-AI have unavailability values of $4.16‰$, $4.10‰$, and $2.81‰$, respectively. Fig. 9 shows the CDF of the unavailability of PMs. It is easy to observe that by using our proposed algorithm, the unavailability of all PMs is less than $6‰$. In comparison, the unavailability of PMs using other benchmarks can be larger than $10‰$. The reason is that RD+PD will take into account the tenant abnormal rate during VM deployment and minimize the impact of the tenant abnormal rate on PM availability.

The fifth set of simulations compares the number of PMs that a tenant can affect by changing the number of tenants in clouds. As shown in Fig. 10, with the increasing number of tenants, the maximum number of PMs that a tenant can affect increases for all algorithms in both topologies, and RD+PD can affect fewer PMs than other algorithms. For example, when there are 2000 tenants in the large topology, the results of NOVA, DSR, R-VMP-AI and RD+PD are 100, 98, 40 and 32, respectively. That means, RD+PD reduces the maximum number of PMs a tenant can affect by 68%, 67.3% and 20% compared with NOVA, DSR and R-VMP-AI, respectively. Figure 11 further illustrates that both RD+PD and R-VMP-AI ensure that the number of affected PMs remains below 40.

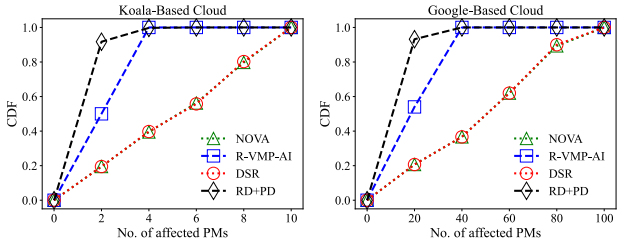

Fig. 11. CDF of the number of PMs that a tenant can affect.

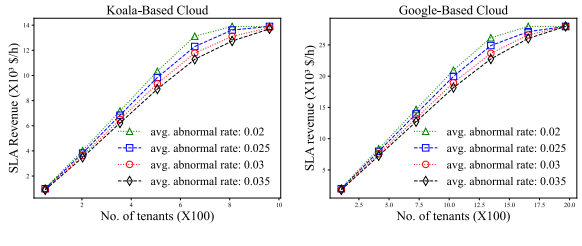

Fig. 12. SLA revenue of CSP under different tenants abnormal rate.

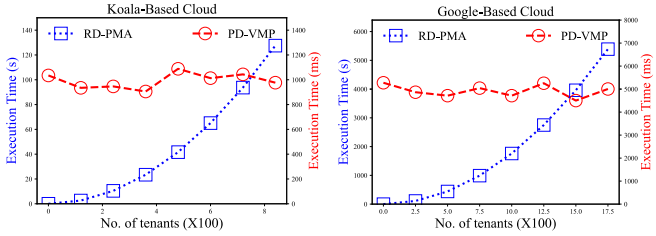

Fig. 13. Execution time vs. Number of tenants.

In contrast, other benchmarks result in more than 80 affected PMs. The reason behind these results is that RD+PD takes into account the impact of tenant uncertainty to ensure the service availability, the number of tenants deployed on each PM is not excessive. R-VMP-AI adopts a similar approach to alleviate the impact of abnormal events in the cloud by setting thresholds.

In the sixth set of simulations, we delve into the sensitivity analysis of our proposed approach, which is a fundamental step in understanding the impact of input data variations on the model's optimal value [43]. Specifically, we focus on the abnormal rate of tenants in the cloud, a crucial parameter beyond the control of the CSP. Building upon previous works [43], [44], we conduct sensitivity analysis by adjusting this parameter. We vary the average abnormal tenant rate and observe the evolution of the CSP's SLA profit as the number of tenants increases, as shown in Fig. 12. The experimental results demonstrate that our method ensures a relatively stable SLA profit for the CSP, even when faced with fluctuations in tenant anomaly rates. For example, in the Google-Based Cloud with 1600 tenants, varying average abnormal rates of 0.02%, 0.025%, 0.003%, and 0.035% respectively, yield SLA profits of $27.96k$, $27.12k$, $26.60k$, and $26.04k\$/h$. These findings underscore the effectiveness of our approach in addressing scenarios involving malicious tenants in the cloud.

In the final set of simulations, we examine the execution time of our proposed approach, as illustrated in Fig. 13. The PD-VMP algorithm demonstrates remarkable problem-solving capabilities by providing suitable VM placement solutions within seconds. As for the RD-PMA algorithm,

in a Koala-Based Cloud with 960 tenants, the algorithm completes within 128s. Similarly, in a Google-Based Cloud with 2,000 tenants, the completion time is 5,389s. It is worth mentioning that in Step 1 of the RD-PMA algorithm, we utilize the Cplex solver [29] with default settings on a small server to solve the linear programming (LP) problem. Consequently, a significant portion of the execution time of the RD-PMA algorithm is allocated to Step 1. For instance, in the Google-Based Cloud scenario, this task consumes over 5,200s. In large-scale commercial clouds, significant optimization can be achieved through techniques such as parallelization on multiple cores. This enables the solution of LP problems with tens of thousands of variables and constraints, similar to the scale of our Google-Based Cloud scenario, within a few minutes [45], [46]. Considering that RD-PMA operates over long-term intervals and has low real-time performance requirements, this timeframe is acceptable.

C. Testbed Evaluation

1) Testbed Settings: To better evaluate RD+PD, we choose OpenStack [33], a widely used cloud infrastructure software, to implement our small-scale testbed. Specifically, we deploy the latest version of OpenStack called Zed [47] on a cluster of 15 PMs. Each PM is equipped with Ubuntu 22.04 OS, two AMD Ryzen 9 3950X CPUs (16)-core, 32-thread) [48], 256GB of RAM, and a 1Gbps network bandwidth. In our experimental setup, we initially deploy VM requests for 13 normal tenants, with each tenant randomly generating 1-10 VM placement requests from Table II. We use iPerf3 [34] to generate service requests of tenants. Then we deploy an abnormal tenant who generates 9 VM placement requests by default in the cloud. Once the VM placement phase is complete, the abnormal tenant initiates DoS attacks using hping3 [49]. We assess the impact of the DoS attacks by measuring the number of affected PMs and tenants. Additionally, we collect data on the round-trip time (RTT) and packet loss rate of each PM, as well as the flow completion time (FCT) for each tenant. It is worth noting that for R-VMP-AI, following the suggestion of Zhao et al. [16], we restrict the number of tenants served by each PM to a maximum of 6, with each tenant's VM being deployed on no more than 3 PMs.

2) Testbed Results: We run three sets of experiments to evaluate the performance of our proposed algorithm and benchmarks. In the first set of experiments, we observe the number of affected PMs and tenants by changing the number of VMs of the abnormal tenant. It is obvious that RD+PD can achieve the best performance among all algorithms. As shown in Fig. 14, when the abnormal tenant deploys 9 VMs, the number of affected PMs by adopting RD+PD is 1. While using NOVA, R-VMP-AI and DSR, the number are 8, 3 and 8, respectively. That means, RD+PD can reduce the number of affected PMs by 87.5%, 66.7% and 87.5% compared with NOVA, R-VMP-AI and DSR, respectively. In Fig. 15, when the abnormal tenant deploys 9 VMs, the number of affected tenants is 2 by adopting our proposed algorithm. While using NOVA, R-VMP-AI and DSR, the number are 12, 7 and 13, respectively. In other words, RD+PD can reduce the number of affected tenants by 83.3%, 71.4% and 84.6% compared with NOVA, R-VMP-AI and DSR, respectively. The reason is that RD+PD takes into account the impact of the tenant uncertainty

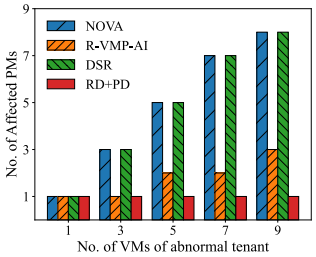

Fig. 14. Number of affected PMs vs. Number of VMs of abnormal tenant.

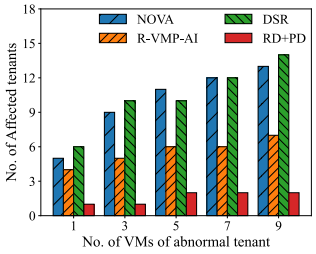

Fig. 15. Number of affected tenants vs. Number of VMs of abnormal tenant.

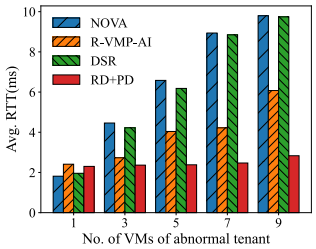

Fig. 16. Average RTT vs. Number of VMs of abnormal tenant.

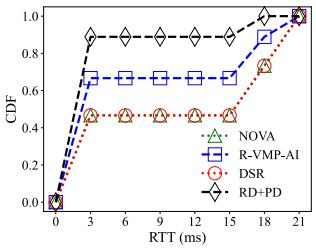

Fig. 17. CDF of the RTT.

on the service availability, so the number of tenants deployed on each PM is not excessive.

The second set of experiments measures the performance of each PM in the cloud when an abnormal tenant launches DoS attacks. In Fig. 16, we demonstrate the average RTT of traffic on each PM by changing the number of VMs of the abnormal tenant. We observe that RD+PD always achieves the least average RTT among all comparison algorithms when the number of VMs of abnormal tenant is larger than 1. For example, when the abnormal tenant deploys 9 VMs, the average RTT are 9.9ms, 6.1ms, 9.8ms and 2.5ms corresponding to NOVA, R-VMP-AI, DSR and RD+PD, respectively. That means, RD+PD reduces the average RTT by 74.7%, 59.0% and 74.4% compared with NOVA, R-VMP-AI and DSR, respectively. Fig. 17 shows the CDF of RTT when the number of VMs of abnormal tenant is 9. We obtain that the proportion of PMs with RTT below 3ms is 46.7%, 66.7%, 46.7% and 86.7% corresponding to NOVA, R-VMP-AI, DSR

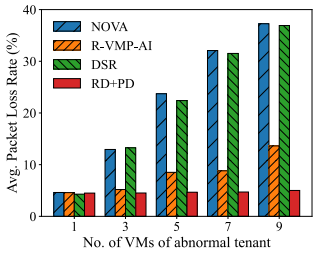

Fig. 18. Average packet loss rate vs. Number of VMs of abnormal tenant.

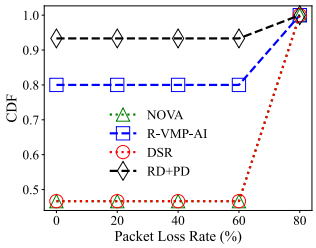

Fig. 19. CDF of the packet loss rate.

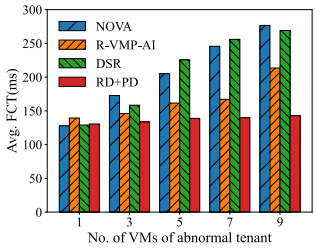

Fig. 20. Average FCT vs. Number of VMs of abnormal tenant.

and RD+PD, respectively. As shown in Fig. 18, we measure the packet loss rate for traffic on each PM and show the average results. When the number of VMs of abnormal tenant is 9, RD+PD can reduce the average packet loss rate by 86.8%, 63.4% and 86.1% compared with NOVA, R-VMP-AI and DSR, respectively. Fig. 19 shows the CDF of the packet loss rate when the number of VMs of abnormal tenant is 9. It is clear that RD+PD can achieve the best performance among the four algorithms since 93.3% of PMs never suffer packet loss. From the above experimental results, we know that RD+PD can control the number of PMs that a tenant can deploy. Therefore, when an abnormal tenant sends abnormal traffic, RD+PD can limit the number of affected PMs and achieve better network performance (*e.g.*, smaller RTT and lower packet loss rate).

Finally, we measure the FCT of each tenant since all tenants expect their traffic to be handled as soon as possible. To measure the FCT performance, each tenant generates 100 flows and sends them to their VM instances. These flows satisfy the 2/8 distribution, of which 20 flows are elephant flows, accounting for 80% traffic, and the remaining 80 mice flows accont for 20% traffic [50]. In Fig. 20, when the abnormal tenant deploys 9 VMs, RD+PD can reduce the average FCT by 47.9%, 32.6% and 46.3% compared with NOVA, R-VMP-AI and DSR, respectively. Fig. 21 shows the CDF of FCT when the number of VMs of abnormal tenant is 9. We obtain the proportion of flows with FCT below 200ms is 42.8%, 66.7%, 42.8% and 92.8% corresponding to NOVA, R-VMP-AI, DSR and RD+PD, respectively. This is because the number of PMs attacked by the abnormal tenant is reduced in RD+PD, consequently, the number of affected flows also decreases.

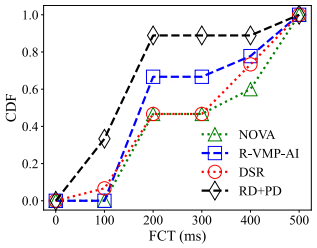

Fig. 21. CDF of the FCT.

VII. Conclusion

This paper considers the service availability in terms of both the hardware availability and the tenant uncertainty, and studies the problem of service availability-guaranteed VM placement in multi-tenant clouds (SAG-VMP). To efficiently solve this complex problem, we take a two-phase approach: PM assignment and VM placement. Two algorithms with bounded approximation factors have been designed to for these two phases, respectively. Both small-scale experiment results and large-scale simulation results show the high efficiency of our proposed algorithms.

Appendix A
Discussion

A. Energy Consumption

Energy consumption is a crucial issue of cloud computing. We employ two approaches to effectively reduce the energy consumption of data centers:

1) Our primary objective is to maximize CSP revenue by optimizing the utilization of PM resources. By efficiently utilizing data center PMs, we can minimize the number of PMs required for service deployment, leading to a reduction in overall energy consumption.
2) Our paper effectively ensures the service availability of tenants by comprehensively considering hardware availability and tenant uncertainty. By prioritizing service availability, our solution can effectively reduce the frequency of VM migration in data centers caused by service unavailable, ultimately reducing energy consumption in the data center.

B. Live Migration

Our proposed method supports live migration operations. For example, when there is a need to migrate VM v of tenant t from the current PM p to another PM, our method involves two steps:

1) Migration Decision: In this step, we create a new VM called v^* with the same specifications as the VM v on another PM as the migration target. This decision is made by running a modified PD-VMP algorithm. Specifically, we update Line 8 of the PD-VMP algorithm from $p^* \leftarrow \arg\min_{p \in P_t} K_p$ to $p^* \leftarrow \arg\min_{p \in \{P_t - \{p\}\}} K_p$. Then, we execute Step 2 of the PD-VMP algorithm to

determine the optimal VM migration target p^*. Finally, we create VM v^* on PM p^*.

2) Migration Execution: In this step, we synchronize the configuration and data of v to v^*. Once synchronization is complete, we perform configuration and data consistency verification. After successful verification, we migrate the tenant's business traffic from v to v^* and delete the VM v.

C. Handling PM Crash and Loss

When a PM crashes while running VMs, it may destroy VMs and potentially violate the tenants' SLA. We address this issue by migrating the affected VMs to other available PMs within the data center. The process of VM migration is elaborated in Appendix A-B. It is worth noting that our proposed algorithm serves as a complementary solution based on existing methods. Therefore, it can be integrated with existing PM failure response strategies, such as the failure prediction technique in [14], to minimize disruptions and ensure a seamless transition, thereby reducing the impact on the SLA for tenants.

If the system detects the loss of PM p during runtime, we update the set of feasible PMs $P_t = P_t - \{p\}, \forall t \in T_r$. This ensures that subsequent VM deployments are not placed on PM p. Additionally, we report the PM loss information to the management plane, which triggers the migration process for the VMs on PM p.

Appendix B
Proof Details

A. Proof of Theorem 2

In this section, we give the upper bound of the Algorithm 1 that may violate the constraints, analyze the approximate ratio based on the randomized rounding method [16], [30]. As preliminary knowledge, we give a famous theorem for approximate performance analysis.

Theorem 8 (Chernoff Bound): Given n independent variables: $a_1, a_2, \ldots, a_n$, where $\forall a_i \in [0,1]$. Let $\kappa = \mathbb{E}\left[\sum_{i=1}^{n} a_i\right]$. Then, we have $\mathbf{Pr}[\sum_{i=1}^{n} a_i \geq (1+\epsilon)\kappa] \leq e^{\frac{-\epsilon^2\kappa}{2+\epsilon}}$ and $\mathbf{Pr}[\sum_{i=1}^{n} a_i \leq (1-\sigma)\kappa] \leq e^{-\sigma\kappa/2}$, where ϵ and σ are arbitrary positive values.

Lemma 9: In RD-PMA, we have $\mathbb{E}[\widehat{z}_t] = \widetilde{z}_t$, $\mathbb{E}[\widehat{y}_t^p] = \widetilde{y}_t^p$ and $\mathbb{E}[\widehat{x}_t^p] = \widetilde{x}_t^p$.

Proof: According to the operation in the second step of KR-PD, we have $\mathbf{Pr}[\widehat{z}_t = 1] = \widetilde{z}_t, \forall t \in T$. Obviously, it follows that:

$$\mathbb{E}[\widehat{z}_t] = \mathbf{Pr}[\widehat{z}_t = 1] \cdot 1 + \mathbf{Pr}[\widehat{z}_t = 0] \cdot 0 = \widetilde{z}_t \tag{12}$$

Based on the definition of condition probability, it is easy to see that:

$$\begin{aligned}\mathbf{Pr}[\widehat{y}_t^p = 1] = &\ \mathbf{Pr}[\widehat{y}_t^p = 1 | \widehat{z}_t = 1]\,\mathbf{Pr}[\widehat{z}_t = 1] \\ &+ \mathbf{Pr}[\widehat{y}_t^p = 1 | \widehat{z}_t = 0]\,\mathbf{Pr}[\widehat{z}_t = 0] \\ = &\ \frac{\widetilde{y}_t^p}{\widetilde{z}_t} \cdot \widetilde{z}_t = \widetilde{y}_t^p, \ \forall t \in T, p \in P\end{aligned} \tag{13}$$

Similar to Eq. (12), we obtain:

$$\mathbb{E}[\widehat{y}_t^p] = \mathbf{Pr}[\widehat{y}_t^p = 1] \cdot 1 + \mathbf{Pr}[\widehat{y}_t^p = 0] \cdot 0 = \widetilde{y}_t^p$$

$$\mathbb{E}[\widehat{x}_t^p] = \mathbf{Pr}[\widehat{y}_t^p = 1] \cdot \frac{\widetilde{x}_t^p}{\widetilde{y}_t^p} + \mathbf{Pr}[\widehat{y}_t^p = 0] \cdot 0 = \widetilde{y}_t^p \cdot \frac{\widetilde{x}_t^p}{\widetilde{y}_t^p} = \widetilde{x}_t^p \tag{14}$$

□

Assume that the minimum resource capacity of all PMs is R_p^{min}. We define a variable γ as follows:

$$\gamma = \min\left\{\frac{R_p^{min}}{D_t}, t \in T\right\} \tag{15}$$

Theorem 10: RD-PMA can achieve the approximation factor of $\frac{\log n}{\gamma} + 1$ for PM resource constraints in a data center, where n denotes the number of PMs.

Proof: Using the randomized rounding method, for each tenant $t \in T_r$, CSP will determine whether each PM $p \in P$ provides services to the tenant t ($\widehat{y}_t^p = 1$) or not ($\widehat{y}_t^p = 0$). We use a variable ϕ_t^p to denote the PM resource consumption when the tenant $t \in T$ is served by PM $p \in P$.

$$\phi_t^p = \begin{cases} D_t, & \text{with probability of } \widetilde{x}_t^p \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

According to the definition, $\{\phi_t^p\}$ are mutually independent. Combining Lemma 9, the resource consumption for one PM node p is:

$$\mathbb{E}\left[\sum_{t \in T} \phi_t^p\right] = \sum_{t \in T} \mathbb{E}[\widetilde{x}_t^p \cdot D_t] \leq R_p \tag{17}$$

Combining Eq. (17) and the definition of γ in Eq. (15), we have:

$$\begin{cases} \dfrac{\phi_t^p \cdot \gamma}{R_p} \in [0,1] \\ \mathbb{E}\left[\displaystyle\sum_{t \in T} \frac{\phi_t^p \cdot \gamma}{R_p}\right] \leq \gamma \end{cases} \tag{18}$$

Then, by applying Chernoff Bound, assume that ϵ is an arbitrary positive value. It follows:

$$\begin{aligned}&\mathbf{Pr}\left[\sum_{t \in T} \frac{\phi_t^p \cdot \gamma}{R_p} \geq (1+\epsilon) \cdot \gamma\right] \leq e^{\frac{-\epsilon^2\gamma}{2+\epsilon}} \Rightarrow \\ &\mathbf{Pr}\left[\sum_{t \in T} \frac{\phi_t^p}{R_p} \geq (1+\epsilon)\right] \leq e^{\frac{-\epsilon^2\gamma}{2+\epsilon}} \leq \frac{1}{n}\end{aligned} \tag{19}$$

where n denotes the number of PM nodes, and $\frac{1}{n}$ is a value close to zero. By solving the above inequality, we have

$$\epsilon \geq \frac{\log n + \sqrt{\log^2 n + 8\gamma \log n}}{2\gamma} \Rightarrow \epsilon \geq \frac{\log n}{\gamma} \tag{20}$$

Thus, the approximate factor for PM resource constraints is $\epsilon + 1 = \frac{\log n}{\gamma} + 1$.

□

Assume that the minimum hardware availability of all PMs is denoted by A_p^{min}. We define a variable τ as follows:

$$\tau = \min\left\{\frac{A_p^{min}}{\mu_t}, t \in T\right\} \tag{21}$$

Theorem 11: RD-PMA can achieve the approximation factor of $\frac{\log n}{\tau}+1$ for service availability constraints in a data center, where n denotes the number of PMs.

Proof: Combining the fourth and fifth inequalities in Eq. 4, we have $\sum_{t\in T} y_t^p \cdot \mu_t \le A_p$. We use a variable ψ_t^p to denote the impact of PM availability due to tenant uncertainty when the tenant $t \in T$ is served by PM $p \in P$.

$$\psi_t^p = \begin{cases} \mu_t, & \text{with probability of } \widetilde{y}_t^p \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

According to the definition, $\{\psi_t^p\}$ are mutually independent. Combining Lemma 9, the availability due to tenant uncertainty for one PM p is:

$$\mathbb{E}\left[\sum_{t\in T}\psi_t^p\right] = \sum_{t\in T}\mathbb{E}\left[\widetilde{y}_t^p \cdot \mu_t\right] \le A_p \tag{23}$$

Combining Eq. (23) and the definition in Eq. (15), we have:

$$\begin{cases} \dfrac{\psi_t^p \cdot \tau}{A_p} \in [0,1] \\ \mathbb{E}\left[\displaystyle\sum_{t\in T}\frac{\psi_t^p \cdot \tau}{A_p}\right] \le \tau \end{cases} \tag{24}$$

Then, similar to Eqs. (19)-(20), we can conclude that the approximate factor of the service availability for VMs on PM p is $\frac{\log n}{\tau}+1$. □

Theorem 12: Suppose O_{LP} is the optimal objective value to the relaxed version of Eq. (4), while O_R is the objective value of Eq. (4) associated with the RD-PMA algorithm. We have $\mathbf{Pr}[O_R \le (1-\sigma)O_{LP}] \le e^{-\sigma O_{LP}/2}$, which means the objective value O_R derived by RD-PMA is close to the optimal value O_{LP} with a high probability.

Proof: From the analysis in Lemma 9, we have $\mathbb{E}[\widehat{z}_t] = \widetilde{z}_t$. Accordingly,

$$\mathbb{E}[O_R] = \sum_{t\in T}\widehat{z}_t \cdot Q_t = \sum_{t\in T}\widetilde{z}_t \cdot Q_t = O_{LP} \tag{25}$$

Then, we have $e^{-\sigma\mathbb{E}[O_R]/2} = e^{-\sigma O_{LP}/2}$. Based on Theorem 8, we have:

$$\mathbf{Pr}[O_R \le (1-\sigma)\mathbb{E}[O_R]] \le e^{-\sigma\mathbb{E}[O_R]/2} \tag{26}$$

Combining the above discussions, we conclude that:

$$\mathbf{Pr}[O_R \le (1-\sigma)O_{LP}] \le e^{-\sigma O_{LP}/2} \tag{27}$$

Thus, the objective value O_R derived by RD-PMA is close to the optimal value O_{LP} with a high probability. □

Approximation Factors: From the above theorems, RD-PMA can achieve the approximation factor of $\frac{\log n}{\gamma}+1$ for the PM resource constraint in a data center. Meanwhile, the service availability constraint will hardly be violated by a factor of $\frac{\log n}{\tau}+1$. It means that the RD-PMA algorithm can achieve the optimal solution, violating the PM capacity by at most a factor $\frac{\log n}{\gamma}+1$, violating the service availability of tenant by at most a factor $\frac{\log n}{\tau}+1$. Thus, we can conclude that RD-PMA can achieve the bi-criteria approximation of $\left(\frac{\log n}{\gamma}+1, \frac{\log n}{\tau}+1\right)$.

B. Proof of Theorem 6

In the following, we will prove the competitive ratio of PD-VMP is $[1/(1+\rho), O(\log J + \log(1/\rho))]$.

Lemma 13: The optimization objective of our proposed PD-VMP algorithm is at least $1/(1+\rho)$ times OPT, where OPT is the result of the optimal solution.

Proof: When a VM v of tenant t is placed on PM p^*, the objective value of the linear program increases by Q_t^v. To avoid the confusion, let $\alpha'(t,v)$ and $\beta'(p)$ denote the values of $\alpha(t,v)$ and $\beta(p)$ after the VM v of tenant t is placed, respectively.

According to the update rules of dual variables in Algorithm 2, the objective value of the dual program increases by Δ:

$$\begin{aligned}
\Delta &= \sum_{v\in V_t}\sum_{t\in T_r}(\alpha'(t,v)-\alpha(t,v)) + \sum_{p\in P} R_p\cdot(\beta'(p)-\beta(p)) \\
&\le \alpha'(t,v) + (\beta'(p^*)-\beta(p^*))\cdot R_{p^*} \\
&= Q_t^v\,(1-K_{p^*}) + R_{p^*}\left\{\left[\beta(p^*)\left(1+\frac{D_t^v}{R_{p^*}}\right)+\frac{D_t^v}{\varphi\cdot R_{p^*}}\right]-\beta(p^*)\right\} \\
&= Q_t^v\left(1-\frac{D_t^v}{Q_t^v}\cdot\beta(p^*)\right)+\left(\frac{\beta(p^*)\cdot D_t^v}{R_{p^*}}+\frac{D_t^v}{\varphi\cdot R_{p^*}}\right)R_{p^*} \\
&= Q_t^v + \frac{D_t^v}{\varphi} = Q_t^v + \frac{D_t^v\rho}{J} \\
&\le (1+\rho)Q_t^v
\end{aligned} \tag{28}$$

The second inequality holds because for each VM request, it is placed on only one PM. The last inequality holds due to the definition of J in Eq. (8). In conclusion, our algorithm increases the objective of the dual algorithm by at most $(1+\rho)Q_t^v$. As a result, the overall objective value of the dual program is at least $1/(1+\rho)$ times as that of the optimal solution. □

In the following, we consider the violation extent of the resource constraint on each VM placement request. For ease of presentation, Let $V = \cup_{t\in T}V_t$, and $v_k (k \le |V|)$ denote the k-th VM placement request. In addition, let $G(p,k)$ and $\beta(p,k)$ denote the load on PM p and the value of $\beta(p)$ after VM placement request v_k has been processed, respectively.

Lemma 14: For each VM placement request v_k, and its assigned PM p, we have

$$\beta(p,k) \ge \frac{\exp\left[G(p,k)/R_p\right]-1}{\varphi} \tag{29}$$

Proof: We prove the lemma by the induction of request $v_k, k = 1,2,\ldots,|V|$. In the initial stage of the algorithm, $\beta(p,0) = G(p,0) = 0$ for all PM p. Thus, the inequalities hold. Note that the value of $\beta(p,k)$ will be updated during the running of Algorithm 2. For each arrival request v_k, if v_k is rejected, two variables $G(p,k)$ and $\beta(p,k)$ will not be updated, *i.e.*, $G(p,k) = G(p,k-1)$ and $\beta(p,k) = \beta(p,k-1)$. Thus the inequality holds as well. If request v_k is accepted, we have $G(p,k) = G(p,k-1) + D_t^v$. According to the update rule of $\beta(p,k)$ in Eq. (10), we also have

$$\beta(p,k) = \beta(p,k-1)\left(1+\frac{D_t^v}{R_p}\right)+\frac{D_t^v}{\varphi\cdot R_p} \tag{30}$$

Based on induction hypothesis, we apply inequality $\beta(p,k-1) \geq \frac{\exp[G(p,k-1)/R_p]-1}{\varphi}$ to Eq. (15) and obtain:

$$\begin{aligned}\beta(p,k) &\geq \frac{\exp\left(G(p,k-1)/R_p\right)-1}{\varphi}\left(1+\frac{D_t^v}{R_p}\right)+\frac{D_t^v}{\varphi\cdot R_p}\\ &= \frac{1}{\varphi}\left[\exp\left(\frac{G(p,k-1)}{R_p}\right)\left(1+\frac{D_t^v}{R_p}\right)-1\right]\\ &\approx \frac{1}{\varphi}\left[\exp\left(\frac{G(p,k-1)}{R_p}\right)\exp\left(\frac{D_t^v}{R_p}\right)-1\right]\\ &= \frac{\exp\left(G(p,k)/R_p\right)-1}{\varphi} \end{aligned} \tag{31}$$

Here we apply the first order approximation, $exp(x) \approx 1+x$ for a small positive value x. Strict inequality can be established by a more complicated update rule and incurs unnecessary complexity. As a result, this lemma holds. □

The following lemma guarantees the performance of PD-VMP in terms of the resource constraint on each PM node.

Lemma 15: The proposed PD-VMP algorithm will not violate the resource constraint by a factor of $O\left(\log J + \log(1/\rho)\right)$ on each PM node.

Proof: According to Algorithm 2, the value of β_p will be updated only if $K_{p^*} \leq 1$ (the request will be accepted in Line 9 of Algorithm 2). Combinning the Eq. (7), we observe that if a PM p satisfies $\beta(p) > 1$, the VM placement request v will be rejected on this PM. This means that before the last update of $\beta(p)$, we have $\beta(p) \leq 1$. According to the update rule of $\beta(p)$ in Line 12 of Algorithm 2 and the definition of J in Eq. (8), we obtain

$$\beta(p) \leq 1 + \frac{D_t^v}{R_p} + \frac{D_t^v}{\varphi\cdot R_p} \leq 1 + 2J \tag{32}$$

By Eq. (29) in Lemma 14, we have

$$\frac{G(p,k)}{R_p} \leq \log\left((1+2J)\cdot\varphi+1\right) = O\left(\log J + \log\frac{1}{\rho}\right) \tag{33}$$

□

Combining Lemmas 13 and 15, we prove the proposed PD-VMP algorithm can achieve the competitive ratio of $[1/(1+\rho), O\left(\log J + \log(1/\rho)\right)]$, where ρ is an arbitrary parameter with $\rho \in (0,1)$, and J is a system dependent constant.

REFERENCES

[1] J. Wang, G. Zhao, H. Xu, H. Huang, L. Luo, and Y. Yang, "Robust service mapping in multi-tenant clouds," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.

[2] *Amazon Web Services*. Accessed: Aug. 1, 2023. [Online]. Available: https://aws.amazon.com/

[3] *Google Cloud Platform*. Accessed: Aug. 1, 2023. [Online]. Available: https://cloud.google.com/

[4] V. Narasayya and S. Chaudhuri, "Multi-tenant cloud data services: State-of-the-art, challenges and opportunities," in *Proc. Int. Conf. Manage. Data*, Jun. 2022, pp. 2465–2473.

[5] *Amazon SLA*. Accessed: Aug. 1, 2023. [Online]. Available: https://www.amazonaws.cn/en/ec2/sla/beijing/

[6] *Google Compute SLA*. Accessed: Aug. 1, 2023. [Online]. Available: https://cloud.google.com/

[7] S. Chhabra and A. K. Singh, "Optimal VM placement model for load balancing in cloud data centers," in *Proc. 7th Int. Conf. Smart Comput. Commun. (ICSCC)*, Jun. 2019, pp. 1–5.

[8] J. Kumar, A. K. Singh, and A. Mohan, "Resource-efficient load-balancing framework for cloud data center networks," *ETRI J.*, vol. 43, no. 1, pp. 53–63, Feb. 2021.

[9] W. Yao, Z. Wang, Y. Hou, X. Zhu, X. Li, and Y. Xia, "An energy-efficient load balance strategy based on virtual machine consolidation in cloud environment," *Future Gener. Comput. Syst.*, vol. 146, pp. 222–233, Sep. 2023.

[10] S. Farzai, M. H. Shirvani, and M. Rabbani, "Communication-aware traffic stream optimization for virtual machine placement in cloud datacenters with VL2 topology," *J. Adv. Comput. Res.*, vol. 11, no. 3, pp. 1–21, 2020.

[11] H. Xing, J. Zhu, R. Qu, P. Dai, S. Luo, and M. A. Iqbal, "An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing," *Swarm Evol. Comput.*, vol. 68, Feb. 2022, Art. no. 101012.

[12] T. Abbasi-Khazaei and M. H. Rezvani, "Energy-aware and carbon-efficient VM placement optimization in cloud datacenters using evolutionary computing methods," *Soft Comput.*, vol. 26, no. 18, pp. 9287–9322, Sep. 2022.

[13] S. Banerjee, S. Roy, and S. Khatua, "Game theory based energy-aware virtual machine placement towards improving resource efficiency in homogeneous cloud data center," in *Proc. IEEE Calcutta Conf. (CALCON)*, Dec. 2022, pp. 293–298.

[14] Q. Lin et al., "Predicting node failure in cloud service systems," in *Proc. 2018 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 480–490.

[15] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable virtual machine placement and routing in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2965–2978, Oct. 2017.

[16] G. Zhao, J. Liu, Y. Zhai, H. Xu, and H. He, "Alleviating the impact of abnormal events through multi-constrained VM placement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 34, no. 5, pp. 1508–1523, May 2023.

[17] X. Liu, B. Cheng, and S. Wang, "Availability-aware and energy-efficient virtual cluster allocation based on multi-objective optimization in cloud datacenters," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 2, pp. 972–985, Jun. 2020.

[18] H. Jia et al., "Security strategy for virtual machine allocation in cloud computing," *Proc. Comput. Sci.*, vol. 147, pp. 140–144, Jan. 2019.

[19] A. O. F. Atya, Z. Qian, S. V. Krishnamurthy, T. La Porta, P. McDaniel, and L. Marvel, "Malicious co-residency on the cloud: Attacks and defense," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2017, pp. 1–9.

[20] *Google: Cluster-Data*. Accessed: Aug. 1, 2023. [Online]. Available: https://github.com/google/cluster-data

[21] S. Yang, F. Li, R. Yahyapour, and X. Fu, "Delay-sensitive and availability-aware virtual network function scheduling for NFV," *IEEE Trans. Services Comput.*, vol. 15, no. 1, pp. 188–201, Jan. 2022.

[22] A. Wood, "Availability modeling," *IEEE Circuits Devices Mag.*, vol. 10, no. 3, pp. 22–27, May 1994.

[23] W. E. Smith, K. S. Trivedi, L. A. Tomek, and J. Ackaret, "Availability analysis of blade server systems," *IBM Syst. J.*, vol. 47, no. 4, pp. 621–640, 2008.

[24] P. Zhang, J. Xu, H. Muazu, and W. Mao, "Access control research on data security in cloud computing," in *Proc. IEEE 16th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2015, pp. 873–877.

[25] S. Li, J. Huang, and B. Cheng, "A price-incentive resource auction mechanism balancing the interests between users and cloud service provider," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 2030–2045, Jun. 2021.

[26] S. K. Pande, S. K. Panda, and S. Das, "A revenue-based service management algorithm for vehicular cloud computing," in *Proc. 17th Int. Conf. Distrib. Comput. Internet Technol. (ICDCIT)* Bhubaneswar, India: Springer, Jan. 2021, pp. 98–113.

[27] S. Jangiti and V. S. S. Sriram, "Scalable and direct vector bin-packing heuristic based on residual resource ratios for virtual machine placement in cloud data centers," *Comput. Electr. Eng.*, vol. 68, pp. 44–61, May 2018.

[28] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *J. Heuristics*, vol. 4, no. 1, pp. 63–86, Jun. 1998.

[29] *IBM ILOG CPLEX Optimizer*. Accessed: Aug. 1, 2023. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio/cplex-optimizer

[30] H. Xu, H. Huang, S. Chen, G. Zhao, and L. Huang, "Achieving high scalability through hybrid switching in software-defined networking," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 618–632, Feb. 2018.

[31] M. B. Cohen, Y. T. Lee, and Z. Song, "Solving linear programs in the current matrix multiplication time," *J. ACM*, vol. 68, no. 1, pp. 1–39, Feb. 2021.

[32] L. Guo, J. Pang, and A. Walid, "Joint placement and routing of network function chains in data centers," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 612–620.

[33] *Build the Future of Open Infrastructure*. Accessed: Aug. 1, 2023. [Online]. Available: https://openstack.org

[34] *iPerf—The Ultimate Speed Test Tool for TCP, UDP and SCTP*. Accessed: May 23, 2024. [Online]. Available: https://iperf.fr

[35] *Openstack Compute Schedulers*. Accessed: Aug. 1, 2023. [Online]. Available: https://docs.openstack.org/nova/latest/admin/scheduling.html

[36] *Amazon EC2 Instance Types*. Accessed: Aug. 1, 2023. [Online]. Available: https://aws.amazon.com/ec2/instance-types

[37] *Amazon EC2 Spot Instances Pricing*. Accessed: Aug. 1, 2023. [Online]. Available: https://aws.amazon.com/ec2/spot/pricing

[38] B. Edwards, S. Hofmeyr, and S. Forrest, "Hype and heavy tails: A closer look at data breaches," *J. Cybersecur.*, vol. 2, no. 1, pp. 3–14, Dec. 2016.

[39] M. P. Goodridge, S. Lakshminarayana, and C. Few, "Analysis of load-altering attacks against power grids: A rare-event sampling approach," in *Proc. 17th Int. Conf. Probabilistic Methods Appl. Power Syst. (PMAPS)*, Jun. 2022, pp. 1–6.

[40] (2018). *HackReport*. [Online]. Available: https://ttcsirt.gov.tt/documents/hackreport2018.pdf

[41] Y. Zhu, Y. Liang, Q. Zhang, X. Wang, P. Palacharla, and M. Sekiya, "Reliable resource allocation for optically interconnected distributed clouds," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 3301–3306.

[42] J. Kong et al., "Guaranteed-availability network function virtualization with network protection and VNF replication," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[43] S. Abdi, L. PourKarimi, M. Ahmadi, and F. Zargari, "Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds," *Future Gener. Comput. Syst.*, vol. 71, pp. 113–128, Jun. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17301735

[44] A. Naseri, M. Ahmadi, and L. PourKarimi, "Reduction of energy consumption and delay of control packets in software-defined networking," *Sustain. Comput., Informat. Syst.*, vol. 31, Sep. 2021, Art. no. 100574. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537921000652

[45] A. Luppold, D. Oehlert, and H. Falk, "Evaluating the performance of solvers for integer-linear programming," Hamburg Univ. Technol., Inst. Embedded Syst., Germany, Tech. Rep., Nov. 2018. [Online]. Available: https://tore.tuhh.de/entities/publication/85cead4b-1722-494e-895b-50df7599377c

[46] T. Koch, T. Berthold, J. Pedersen, and C. Vanaret, "Progress in mathematical programming solvers from 2001 to 2020," *EURO J. Comput. Optim.*, vol. 10, 2022, Art. no. 100031.

[47] *Openstack Zed Projects*. Accessed: Aug. 1, 2023. [Online]. Available: https://docs.openstack.org/zed/index.html

[48] *AMD RyzenT 9 3950x*. Accessed: Aug. 1, 2023. [Online]. Available: https://www.amd.com/en/products/cpu/amd-ryzen-9-3950x

[49] *Hping—Active Network Security Tool*. Accessed: Aug. 1, 2023. [Online]. Available: http://wiki.hping.org/

[50] L. Li, K. Xie, S. Pei, J. Wen, W. Liang, and G. Xie, "CS-sketch: Compressive sensing enhanced sketch for full traffic measurement," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 2338–2352, May 2024.

Jiawei Liu received the B.S. degree from the College of Computer Science and Technology, Jilin University, in 2014. He is currently pursuing the Eng.D. degree in computer technology with the University of Science and Technology of China. His current research interests include software-defined networks, cloud computing, and programmable networks.

Gongming Zhao (Member, IEEE) received the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2020. He is currently an Associate Professor with the University of Science and Technology of China. His current research interests include cloud computing, software-defined networking, data center networks, and networking for AI.

Hongli Xu (Member, IEEE) received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China (USTC), China, in 2002 and 2007, respectively. He is currently a Professor with the School of Computer Science and Technology, USTC. He has published more than 100 papers in famous journals and conferences, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the International Conference on Computer Communications (INFOCOM), and the 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 Things. He was awarded the Outstanding Youth Science Foundation of NSFC in 2018. He has won the best paper award or the best paper candidate at several famous conferences.

Peng Yang is currently pursuing the Eng.D. degree in computer science with the University of Science and Technology of China. His main research interests include eBPF/XDP technology, data center networks, and distributed training.

Baoqing Wang is currently pursuing the master's degree in computer science with the University of Science and Technology of China. His main research interests include software-defined networks and cloud computing.

Chunming Qiao (Fellow, IEEE) is currently a SUNY Distinguished Professor and also the current Chair of the Department of Computer Science and Engineering, The State University of New York, Buffalo, NY, USA. His current research interests include connected and autonomous vehicles, and quantum networks. He was elected as an IEEE Fellow for his contributions to optical and wireless network architectures and protocols.