Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning

Junjie Sheng†
East China Normal University, China

Lu Wang
Microsoft Research, China

Fangkai Yang
Microsoft Research, China

Bo Qiao
Microsoft Research, China

Hang Dong
Microsoft Research, China

Xiangfeng Wang†
East China Normal University, China

Bo Jin
East China Normal University, China

Jun Wang
East China Normal University, China

Saravan Rajmohan
Microsoft 365, United States

Si Qin
Microsoft Research, China

Qingwei Lin†
Microsoft Research, China

Dongmei Zhang
Microsoft Research, China

ABSTRACT

Oversubscription is a common practice for improving cloud resource utilization. It allows the cloud service provider to sell more resources than the physical limit, assuming not all users would fully utilize the resources simultaneously. However, how to design an oversubscription policy that improves utilization while satisfying some safety constraints remains an open problem. Existing methods and industrial practices are over-conservative, ignoring the coordination of diverse resource usage patterns and probabilistic constraints. To address these two limitations, this paper formulates the oversubscription for cloud as a chance-constrained optimization problem and proposes an effective Chance-Constrained Multi-Agent Reinforcement Learning (C2MARL) method to solve this problem. Specifically, C2MARL reduces the number of constraints by considering their upper bounds and leverages a multi-agent reinforcement learning paradigm to learn a safe and optimal coordination policy. We evaluate our C2MARL on an internal cloud platform and public cloud datasets. Experiments show that our C2MARL outperforms existing methods in improving utilization (20% ~ 86%) under different levels of safety constraints.

CCS CONCEPTS

• Computing methodologies → Artificial intelligence; • Networks → Cloud computing.

†Work done during Junjie’s internship at Microsoft Research.
‡Corresponding author: Xiangfeng Wang (xfwang@sei.ecnu.edu.cn) and Qingwei Lin (qlin@microsoft.com).

KEYWORDS

Cloud Computing, Over Subscription, Reinforcement Learning, Multi-Agent System

ACM Reference Format:
Junjie Sheng, Lu Wang, Fangkai Yang, Bo Qiao, Hang Dong, Xiangfeng Wang, Bo Jin, Jun Wang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning. In Proceedings of the ACM Web Conference 2023 (WWW ’23). April 30–May 04, 2023, Austin, TX, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3543507.3583298

1 INTRODUCTION

Cloud computing has been widely adopted in recent years. Many companies, e.g., Netflix and LinkedIn, request cloud computing resources from cloud providers to host their services and applications [26]. The requested resources are usually served by Virtual Machines (VMs), which are virtualized instances hosted on Physical Machines (PMs). Users (what is called Subscribers in this paper) estimate their resource usage and request a number of VMs accordingly. The actual usage of computing resources is unstable over time in general (colored areas in Figure 1), e.g., email applications have peak resource usage during working hours. Then subscribers take the peak usage for references when requesting computing resources, and the requested resources are usually overestimated [4]. In reality, the actual usage does not always peak to reach the requested resources. For example, the services and applications running on the Alibaba cloud platform have less than 50% actual usage compared to the requested in the majority of time [12]. Therefore, a large amount of requested resources is not utilized, and it leaves an opportunity to reduce such waste and improve resource utilization.

An effective way to reduce unused resources is oversubscription [15, 24]. For each VM on a PM, oversubscription assigns less than the requested resources to the VM and allows the VM to use beyond the assigned resources up to the requested resources when the PM has spare capacity†. Figure 1 shows that oversubscription improves resource utilization by allocating more VMs on the same

†Currently cloud providers only do oversubscription for internal users [1].
Physical Machine (PM). Although oversubscription improves resource utilization, it creates risks if it is done inappropriately and overaggressively. The hot machine problem [22] is one of the risks that impacts server reliability. More specifically, if the sum of the actual usage of these VMs exceeds the resource capacity of the host PM, the PM becomes a hot machine, and it will have a high risk on crash. Thus, the key challenge is to design an oversubscription policy that reduces the most unused resources while having a low probability of causing hot machines.

In industrial practice, grid search [11] is a common oversubscription policy. It searches a static oversubscription rate for all subscribers, i.e., computed by assigned resources over requested resources. For example, Figure 1(b) shows that the grid search oversubscription policy can allocate one more VM to increase resource utilization while avoiding hot machines. However, the grid search approach is often nonadaptive as it ignores the heterogeneity of actual usage patterns among various subscribers, which limits utilization improvement. Moreover, it is impractical to conduct the grid search for each subscriber as the search space is growing exponentially with the number of subscribers. Kumbhare et al. [15] and Chen et al. [5] propose supervised-learning-based oversubscription approaches that estimate the actual or the maximal resource usage for each subscriber (Figure 1(c)). However, existing methods often come up with conservative oversubscription policies for two major reasons: 1) they ignore the coordination of heterogeneous resource usage patterns among subscribers, and 2) they are unable to leverage probabilistic constraints. Figure 2 gives an example of the coordination of two subscribers. With the consideration of heterogeneity of usage patterns of different subscribers, e.g., staggered peaks, the assigned resources can be lower than peak usage without causing hot machines. Thus there is a great opportunity for multi-agent coordination to improve resource utilization. On the other hand, cloud systems are highly stochastic. The usage of a virtual machine is usually hard to predict and there is always a small chance that all the virtual machines reach their peak usage simultaneously. Avoiding the small chance events inevitably limits a lot on achieving benefits and probabilistic constraints are often taken as good practice for that better describe the constraints [18]. The existing methods cannot leverage the probabilistic constraints but rather design oversubscription policies toward fulfilling deterministic safety constraints regardless of their diversity.

To address the above two challenges, this paper formulates the oversubscription problem as a chance-constrained multi-agent optimization problem. In order to coordinate the resource utilization of subscribers under chance constraints, this paper proposes a novel chance-constrained multi-agent reinforcement learning (C2MARL) method. It brings into value decomposition networks to deal with the unknown dynamics of the multi-agent cloud system, and it proposes a primal-dual reformulation to address the non-convex chance constraints. Overall, C2MARL seeks to maximize resource utilization by saving more resource capacity for the coming VMs while constraining the number of hot machines under the threshold with a high probability (Figure 1(d) and 2).

Our contributions can be summarized as follow:

- This paper formulates the oversubscription problem in Cloud as a chance-constrained optimization problem.
- A novel MARL algorithm, C2MARL, is proposed to solve the chance-constrained optimization problem in a large-scale cloud system. To the best of our knowledge, C2MARL is the first method that leverages multi-agent to solve the chance-constrained problem.
- Experiments on both an internal cloud platform and public Azure datasets show that C2MARL achieves better performance (improves 20% ~ 86% in utilization compared with baselines) under different levels of safety constraints.

2 RELATED WORKS

2.1 Oversubscription in cloud

In cloud computing, oversubscription is to assign less than requested resources to the VMs on a PM so that more VMs could be allocated on the PM [14]. Many resources are considered to be oversubscribed, e.g., CPU [6], memory [4], power [15], etc. This paper only considers CPU/core oversubscription as CPU is one of the most scarce resources in the cloud and the primary resource considered...
in VM allocation [13]. The industrial practice applies grid search methods [11] to find a global oversubscription rate that maximizes the saved resources, trying not to violate the constraints. Ghosh and Naik [9] emphasizes the importance of actual usage patterns when designing oversubscription policies. Caglar and Gokhale [3] and Kumbhare et al. [15] adopt Neural Networks to learn the usage pattern of VMs, and Chen et al. [5] leverages lightweight Brown’s Double Exponential Smoothing to predict usage patterns. Other oversubscription works propose new resource allocation methods to improve resource utilization [6]. In reality, the resource allocator is a complex system and consists of numerous rules and preferences to fulfill multiple objectives [13], which limits the feasibility of incorporating these works into the practical resource allocator.

### 2.2 Chance-Constrained Reinforcement Learning

Chance-constrained reinforcement learning has been prevalent in recent years. Some model-based chance-constrained RL methods improve the over-conservative policy with efficient evaluations [20, 21]. However, it is challenging to apply model-based methods due to the complexity of the cloud system. Other model-free chance-constrained RL methods are mainly designed based on penalty methods and Lagrangian methods. Geibel and Wysotzki [8] first adds a large penalty on the reward when the chance constraint is violated. Paternain et al. [19] derives a lower bound of the safe probability and reformulates the chance-constrained RL as a primal-dual optimization problem. However, the oversubscription problem in this paper involves a large number of chance constraints and subscribers, which makes it challenging to leverage existing methods.

### 3 PRELIMINARY

#### 3.1 Chance Constrained Dec-POMDP

This section first introduces the Chance-Constrained Decentralized Partial Observable Markovian Decision Process (C2Dec-POMDP). It can be formally described by a tuple $(I, \mathcal{S}, \{\mathcal{A}_i\}, \mathcal{P}, \{\Omega_i\}, O, R, T, y, \{C_k\}_k, \{\delta_k\}_k, \alpha)$. $I$ is the set of agents, and each agent corresponds to a subscriber in our setting. $\mathcal{S}$ is the state space. $\mathcal{A}_i$ is agent $i$’s action space and $\mathcal{P} := \times_{i \in I} \mathcal{A}_i$ is the joint action space. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function and $\mathcal{P}(s'|s, a)$ is the probability that the state transits to $s'$ when taking joint action $a$ in state $s$. $\Omega_i$ is the agent $i$’s observation space and the $\Omega := \times_{i \in I} \Omega_i$ is the joint observation space. $O : \mathcal{S} \rightarrow \Omega$ is the observation function and $O(\omega|s)$ describes the probability of joint observation $\omega$ when current state is $s$. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ is the reward function and $r(s, a)$ is the immediate reward when taking action $a$ in state $s$. $T$ is the time horizon. $y$ is the discount factor. $C_k$ is the $k$-th constraint and $C_k(s, a)$ is the immediate $k$-th constraint cost. $\delta_k$ is the upper bound on the cumulative $k$-th constraint cost. $\alpha_k$ is the probability that the cumulative constraint cost is less than the $\delta_k$. $\alpha$ is the probability that the constraints need to be satisfied. Each agent $i$ has a policy $\pi_i(a_i|\omega_i)$ that maps $\omega_i$ to the local action $a_i$. The goal of solving the C2Dec-POMDP is to find a policy for each agent that maximizes the cumulative reward while satisfying $K$ chance constraints:

$$\max \pi \hspace{1cm} \mathbb{E}_{\pi} \left[ \sum_{t=0}^{T} r(s_t, a_t) \right]$$

subject to

$$\mathbb{P} \left( \sum_{t=0}^{T} C_k(s_t, a_t) < \delta \right) \geq \alpha, \forall k$$

$$s_{t+1} \sim \mathcal{P}(s_t, a_t); \ a_t \sim \pi_i(\cdot|\omega_i); \ a_t = \mathcal{O}(s_t), \forall t, i$$

### 4 PROBLEM FORMULATION

This section first gives a programming formulation for the oversubscription problem as a chance-constrained optimization problem. Then we bring it to the sequential decision view and reformulate it as a C2Dec-POMDP.

#### 4.1 Programming Formulation

This paper only oversubscribes the CPU resources as the CPU is often the most valuable and scarce resource in VM allocation [13]. It can also be easily adopted on memory oversubscription and oversubscription on other resources without loss of generalization. The cluster has $K$ PMs where each PM $k$ has $B$ resource capacity in CPU cores. There are $N$ subscribers, each subscriber $n$ requests $C_{nk}$ cores and deletes $D_{nk}$ cores at timestamp $t$. Note that deleted cores are caused by VM leaving. We use $E_{nk}^t$ to represent the assigned cores to subscriber $n$ currently allocated onto PM $k$. The oversubscription action $A_{nk}^t$ is taken for subscriber $n$ at timestamp $t$ to give $C_{nk}^t \cap A_{nk}^t$, where $\circ$ is the element-wise product. Our goal is to maximize the remaining cores over all $K$ PMs in the whole time horizon and constrain the actual core usage $U_{nk}^t$ not to exceed the hot PM threshold $\beta B$ for more than $\delta$ times.

$$\max \mathcal{A} \hspace{1cm} \sum_{t=0}^{T} \sum_{k=1}^{K} (B - \sum_{n=1}^{N} E_{nk}^t)$$

subject to

$$\mathbb{P} \left( \sum_{t=0}^{T} \sum_{n=1}^{N} U_{nk}^t \geq \beta B \right) < \delta, \forall n, k$$

$$E_{nk}^t = f(E_{nk}^{t-1}, D_{nk}^{t-1}, C_{nk}^{t-1} \cap A_{nk}^{t-1}),$$

where $f$ can be Best-Fit, First-Fit, Round-Robin, or other scheduler methods. Given previous existing cores, VMs to be created and deleted, and oversubscription actions, $f$ is responsible for allocating oversubscribed VMs to different PMs, giving the new reserved core status $E_{nk}^t$. It also should be noted that the scheduler needs to meet the memory and network bandwidth constraints (only PM with enough memory and network resources can be selected to allocate the VM). Due to the memory and network constraints being handled by the scheduler itself, Eq. (2) does not explicitly describe it for simplicity.

#### 4.2 Markovian Formulation

As the oversubscription action should be taken in an online manner, this section reformulates the chance-constrained programming as the C2Dec-POMDP. Specifically for the oversubscription problem, each subscriber $i$ is denoted as an agent $i$ and its action space $\mathcal{A}_i$ is the oversubscription rate space which ranges from 0 to 1. The state is composed of the information of all PMs and each agent’s current requested resources. The transition probability $\mathcal{P}$ is determined.
by the scheduler function \( f \) as the last constraint in Eq. (2). Each subscriber’s requested resources are scaled based on the oversubscription rate. The cloud platform adopts its scheduler to handle current requests based on oversubscribed resources. Then the state of all PMs is updated and subscribers make new requests, after which new states are transitioned. The reward function is the current remaining cores over all PMs.

**Definition 1** \((\alpha - \delta \text{ PM Safety})\). A Physical Machine \( k \) achieves \(\alpha - \delta \text{ PM safety} \) under the joint policy \( \pi \) if \( \Pr\left(\frac{1}{T}\sum_{t=1}^{T} C_k(s_t) < \delta|\pi\right) \geq \alpha \). The \( C_k \) returns 1 if the PM \( k \) is hot\(^2\) else returns 0.

The constraints for the C2Dec-POMDP are that all the PMs satisfy the \(\alpha - \delta \text{ PM safety} \). The goal of agents for subscribers is to coordinate with each other to maximize cumulative remaining cores (cumulative rewards) without violating the chance constraints.

\[
\max_{\pi} J(\pi) = \mathbb{E}_{s_0} \left[ \sum_{t=0}^{T} r(s_t, a_t) \right]
\]

s.t. \( \Pr\left(\frac{1}{T}\sum_{t=1}^{T} C_k(s_t) < \delta|\pi\right) \geq \alpha, \quad \forall k \) \tag{3}

\( s_{t+1} = p(s_t, a_t); \ a_t^{ij} = \pi_t^{ij}(|a_t|); \ a_T = O(s_T), \forall t, i \)

There are several challenges in solving such a problem: 1) the number of chance constraints increases linearly with the number of PMs, 2) the transition function is not available to the agents, 3) Eq. (3) is a non-convex optimization problem.

5 PROPOSED METHODS: C2MARL

To address the aforementioned challenges, this section proposes a novel Chance-Constrained Multi-Agent Reinforcement Learning (C2MARL) method. This section first reformulates the problem as a cluster-level C2Dec-POMDP problem to address the first challenge. To establish the relationship between policy and chance constraint satisfaction and solve the non-convex optimization, this section further relaxes the problem as a primal-dual multi-agent learning problem. After that, C2MARL is formally proposed to optimize the primal-dual multi-agent learning problem and learn the maximal benefits and safe oversubscription policy for each subscriber.

5.1 Cluster-Level Chance-Constrained Reformulation

Eq. (3) has \( K \) chance constraints and \( K \) is usually a large number in practice. This often makes the optimization difficult [10]. This section transforms Eq. (3) to a cluster-level chance constraint optimization problem that has only one constraint, where the cluster is the aggregated representation of all PMs.

**Definition 2** \((\alpha - \delta \text{ Cluser Safety})\). A cluster has \(\alpha - \delta \text{ safety} \) under joint policy \( \pi \) if \( \Pr\left(\frac{1}{T}\sum_{t=1}^{T} C(s_t) < \delta|\pi\right) \geq \alpha \), where \( C(s_t) \) := \max_k C_k(s_t).

**Theorem 3.** For any policy \( \pi \) that obtains \(\alpha - \delta \text{ cluster safety} \), it has \(\alpha - \delta \text{ PM safety} \) on all the PMs.

\(^2\)The PM is hot if its CPU usage exceeds the hot threshold \( \beta B \).
The reward in each step can be decomposed into two parts: the remaining cores on all PMs in the cluster, and the constraint violation penalty in the cluster. Then the primal problem is transformed into a typical multi-agent reinforcement learning (MARL) problem and many MARL methods can be taken to obtain an oversubscription policy for each subscriber. C2MARL takes a simple and powerful multi-agent reinforcement learning method, VDN [27], as the backbone. Specifically, C2MARL estimates the cumulative rewards after taking joint action $a_i$ in $s_j$ through a state-action value network $Q(s_j, a_i; \theta)$, by $N$ local observation-action value network ($Q_i(\phi_i, \theta_i')$) and a state-value function ($Q_i(\phi_i, \theta_i)$). The intuition behind this is that the cumulative reward is influenced by both the current cluster status and each agent’s action selection. Besides that, only the agent (subscriber) that has requests in the current state, its current action contributes benefits/costs to the cumulative rewards. Thus C2MARL estimates $Q(s_j, a_i; \theta)$ as:

$$Q(s_j, a_i; \theta) = Q^c(s_j^c; \theta^c) + \sum_{i} m_i^c Q_i^c(a_i^c; \theta_i^c),$$

(9)

where $a_i^c$ is the local observation of agent $i$, and $\theta_i$ and $\theta_i'$ are the neural network parameters of the cluster value function and subscriber $i$’s local observation-action-value function, respectively. With the decomposition, the primal problem can be solved accordingly. Specifically, C2MARL follows the common target network design and the double Q learning to stabilize the training. Denote the network parameters as $\theta := [\theta^c, \theta^1, \ldots, \theta^N]$. Define the target parameters $\tilde{\theta} := [\tilde{\theta}^c, \tilde{\theta}^1, \ldots, \tilde{\theta}^N]$ where $\tilde{\theta}^c$ is the cluster value function parameter and $\tilde{\theta}^i$ is the agent $i$’s target state action value function’s parameter. Then the loss function of the primal problem is:

$$L = \arg\min_{\theta} \| Q^c(s_{j+1}^c; \theta^c) + \sum_{i=1}^{N} m_i^c Q_i^c(a_i^c; \theta_i^c) - (r_j + y) \|_2$$

s.t. \hspace{1cm} $y = Q^c(s_{j+1}^c; \theta^c) + \sum_{i=1}^{N} Q_i^c(a_i, a_{i+1}; \theta_i^c, \theta_{i+1}; \tilde{\theta}_i^c)$

(10)

After updating the network parameters $\theta$, the target network parameters are soft updated by $\tilde{\theta} = (1 - \tau) \theta + \tau \tilde{\theta}$, where the $\tau$ is the soft update factor.

The dual problem is a convex problem and can be solved with a closed-form solution. Define the dual learning rate as $\eta_{\lambda}$.

$$\lambda = \max(0, \lambda - \eta_{\lambda}(c - U(\theta^{k+2})))$$

(11)

The details of the algorithm are summarized in Alg 1. Each agent selects its action with $\epsilon$-greedy exploration based on its individual $Q$ network. During the optimization period, the primal parameters, target parameters, and dual parameters are updated sequentially.

6 EXPERIMENTS

This section first proposes a practical oversubscription environment, Oversub Gym, where we can access the approximate state transition by replaying the log data. Then it introduces the baselines and metrics for evaluating our method. After that, we compare our method with baselines in different scenarios.

Algorithm 1 C2MARL: Chance-Constrained Multi-Agent Reinforcement Learning

1. Initialization: the number of over-subscribers $N$, the cluster value network parameters $\theta^c$, each over-subscriber $i$’s state action value network parameters $\theta^i$, discount factor $\gamma$, target network parameters $\tilde{\theta}$, dual learning rate $\eta_{\lambda}$, safety threshold $\alpha$, hot cluster indicator $h^i$.

2. for Episode = 1, $\ldots$, $M$ do

3. Reset state and obtain the initial state $s_1 := [s_1^c, o_1^1, \ldots, o_1^N]$

4. for $t = 1, \ldots, T$ and $s_t \neq$ terminal do

5. for each agent $i$ do

6. With probability $\epsilon$ pick a random action $a_i$ else choose action with the largest value in $Q_i^c(a_i, o_i)$

7. end for

8. Execute global actions and get global reward $r_t$, constraint value $c_t$, next state $s_{t+1} := [s_{t+1}^c, o_{t+1}^1, \ldots, o_{t+1}^N]$

9. Store $(s_t, a_t, r_t, c_t, s_{t+1})$ to $\mathcal{R}$

10. end for

11. for $k = 1, \ldots, K$ do

12. Sample a random mini-batch transitions from $\mathcal{R}$

13. Update $\theta$ by minimizing Eq. 10

14. Update target $Q$ network by soft update manner

15. Update dual parameter by Eq. 11

16. end for

17. end for

6.1 Dataset

This paper takes the internal cloud dataset as one of the backbones of the simulator. It will be released after dropping sensitive information. And another dataset (Azure) is a public and open-sourced dataset. There are 13 subscribers in the internal cloud, while 9 subscribers are in Azure’s cluster.

6.2 Environment Settings

This section presents the design of the Oversub Gym. It contains two environments (internal and public cloud) that have different user patterns [17]. The Oversub Gym takes the traces from an internal cloud computing cluster in February 2022 (denoted as Internal Cloud) and traces from a public cloud (Azure[23]) in 2019 as the data backbone of two environments. Besides that, the Oversub Gym has two modes: cold-start and ward-start, to approximate real industrial scenarios. For the cold-start mode, the environment starts from an empty cluster that has no VMs placed in it. For the warm-start mode, the environment starts from the current cluster status with VMs placed as the log-data shows. The critical designs of the Oversub Gym as shown below.

- **Reset.** For the Cold-start scenario, the Oversub Gym resets the environment with a sufficient number of PMs without VMs running on them. For the Warm-start scenario, it allocates the current running VMs on the corresponding PMs according to the log data.

- **State.** The Oversub Gym takes the current cluster status, the current requests, and the hour of the day as the state.

3https://github.com/Azure/AzurePublicDataset
For the cluster status, we take a coarse representation: \([cpu^{allo}, cpu^{requested}, mem^{allo}, network^{allo}]\). Each component is a \(N\) dimension vector, and each dimension represents each subscriber’s status. The request status is similarly defined as \([cpu^{request}, mem^{request}, network^{request}]\).

**Transition.** The transition model is constructed by the real-world dataset and the allocator. Due to the intractable online training in the real-world cloud, we take the log data to simulate the user requests and usage patterns. For the allocator, best-fit is the major allocation policy in industrial allocators [13], and we use best-fit to simulate the cloud allocator decision.

**Action.** Following the industry practice, we consider discrete action space where each subscriber chooses an oversubscription rate from \([0.2, 0.3, 0.4, 0.5, 0.6, 1]\).

**Reward.** The reward is the number of cores saved by oversubscription \((cpu^{request} - cpu^{assigned})/Z\) where \(Z\) is a large constant for normalizing the reward.

**Constraint.** The constraint is defined as Eq. (2) which requires each PM not hot over 3 times \((\delta = \frac{1}{3})\) with more than \(\alpha\) probability.

### 6.3 Environment Analysis

This section analyzes critical patterns in the cloud and emphasizes the opportunity C2MARL can obtain compared with the baselines.

**Cluster-level Pattern.** As shown in the Figure 3(a), the CPU usage rate (actual resource usage/requested resources) of cluster is range from 0.1 to 0.3. We observe that nearly 70% requested resources are not used in these 5 days. Thus there is an excellent opportunity for oversubscription to reduce wasted resources.

**Subscription Pattern.** Figure 3(b) visualizes three subscribers’ average usage rate patterns. We observe that different subscriber has different usage pattern, which motivates us to set diverse oversubscription rate for different subscribers.

**VM Pattern.** Figure 3(c) shows 5 VMs’ usage pattern. We find that VMs usually have peaks and can be in low duration. The peak usage for a VM is often more significant than the VM’s average usage rate, and it only lasts for a small duration. Moreover, the VMs’ peaks are often staggered.

### 6.4 Metrics and Baselines

Due to the importance of improving the remaining cores while satisfying the chance constraints, we define metrics to reflect the benefit of the remaining cores and safety.

The chance-safety is evaluated via PM-hot ratio (short as PM-Hot-R), whether chance-safety ratio \(\geq V\) (short as \(V\), \(V \in \{0.75, 0.85, 0.95\}\)). PM-Hot-R indicates the maximum probability of a PM in the cluster being hot than \(\delta\).

The remaining cores quantity is evaluated via saved cores (short as S-Cores). S-Cores is the ratio of the saved cores and total request cores (i.e., \(1-\) assigned cores / requested cores). Larger S-Cores indicate better performance. The saved cores are irrelevant to the initial number of PMs and make a clearer comparison than the remaining cores.

Three baselines are considered in this paper: Grid Search [11] is an easy-to-implement policy and is widely used in the industry. It searches for a global and static oversubscription rate for all subscribers. We consider Grid-0.2, -0.4, and -0.6 as baselines as they are the three best oversubscription rates. Supervised oversubscription (short as SL) is proposed in the Kumbhare et al. [15] that predicts the maximum usage rate of each subscriber as the subscriber’s oversubscription rate. Moving average (short as MA) oversubscription [5] adopts the moving average usage rate for each subscriber as the oversubscription policy. More experiment details are presented in the Appendix.

### 6.5 Internal Cloud (Cold Start)

We first conduct experiments on the cold-start cluster by training our model C2MARL with 1800 episodes and 3 different seeds. Three different levels of safety preferences are considered 0.75, 0.85, and 0.95. Figure 4(b) shows the remaining cores of C2MARL during training episodes. It can be observed that: (1) with higher safety preference, C2MARL converges to lower remaining cores. (2) Figure 4(a) visualizes the hot cluster count during learning, where the higher safety preference leads to a less hot cluster count.

We also test our methods and the baselines in different safety preferences setting with 300 episodes, where the comparison results are shown in Table 1. We observe that (1) C2MARL outperforms all the baselines on both chance-safety satisfaction and saved cores with three different safety levels. (2) Grid-0.6, MA, and SL also satisfy all the safety constraints but with sacrificing a large number of S-cores compared with our C2MARL, which demonstrates that chance constraint can help more remaining cores.

| Method       | PM-Hot-R | S-Cores | 0.75 | 0.85 | 0.95 |
|--------------|----------|---------|------|------|------|
| Grid-0.2     | 30.9     | 80      | X    | X    | X    |
| Grid-0.4     | 22.7     | 60.0    | ✓    | ✓    | ✓    |
| Grid-0.6     | 0.0      | 40.0    | ✓    | ✓    | ✓    |
| MA           | 0        | 49.2    | ✓    | ✓    | ✓    |
| SL           | 0.2 ± 0.3| 49.9 ± 3.3| ✓    | ✓    | ✓    |
| Our-0.75     | 9.8 ± 0.6| 66.1 ± 6.7| ✓    | –    | –    |
| Our-0.85     | 6.6 ± 0.6| 61.7 ± 9.2| ✓    | ✓    | –    |
| Our-0.95     | 3.5 ± 0.5| 59.9 ± 1.9| ✓    | ✓    | ✓    |

**Case Study.** To analyze how C2MARL-0.95 outperforms the baselines, we visualize two randomly selected cases in Figure 5 and Figure 6. At 59, two subscribers request VMs, and their VMs’ usage rates peak at different times. SL selects an oversubscription rate that is much larger than the peak usage for each subscriber to mimic the largest usage rate. Our C2MARL takes a more effective action (i.e. 0.2 for the two subscribers) that exploits the staggered peaks via coordination. At the time 77, one subscriber requested VMs with a maximum usage larger than 0.8, SL takes a conservative action (larger than 0.8) while our C2MARL takes 0.6. This does not make the 0.95 chance constraints being violated due to the low duration of the VMs (2 hours) and large variance of the usage. By
Learning Cooperative Oversubscription for Cloud by Chance-Constrained Multi-Agent Reinforcement Learning

Figure 3: CPU usage rate patterns in different levels. The 3(a), 3(b) and 3(c) show the sampled CPU usage rate in the cluster. In 3(a), the cluster’s CPU usage has a daily pattern, and the low usage rate indicates the over-estimation nature of the subscribers. In 3(b), each subscriber has a different CPU usage pattern which brings an opportunity to improve utilization through subscribers’ coordination. 3(b) shows that each VM has a different CPU usage peak and duration.

Figure 4: The training statistics of C2MARL in Internal Cloud with Cold and Warm Start. The line and shadowed area represent the mean quantity and the standard deviation across different runs. The 4(a) and 4(b) depict the hot cluster count and remaining cores of C2MARL with Cold Start, respectively. The 4(c) and 4(d) show the same statistics with Warm Start. In both Cold and Warm Start conditions, a smaller safety threshold results in a higher converged hot cluster count. The remaining cores follow a similar trend in Cold Start but converge to similar values in Warm Start under varying safety thresholds.

6.6 Internal Cloud (Warm Start)

In practice, the oversubscription policy is usually applied to non-empty clusters (Warm Start), where a set of VMs has been placed in the cluster in advance. We extract these pre-allocated VMs from log

exploiting the opportunity of the staggered peaks and low duration of some VMs, our C2MARL obtains an excellent performance on the S-Cores while satisfying the safety constraints.
data that exist before the oversubscription task to initiate the cluster as a warm-start setting. We train C2MARL with 1200 episodes in 3 different seeds, and the results are shown in Figure 4(c) and 4(d). Figure 4(c) visualizes the hot cluster count during learning, where the higher safety preference leads to a less hot cluster count. Figure 4(d) visualizes the remaining cores during learning, where there is a less clear monotonicity relationship between the safe threshold and the remaining cores. The reason would be that the remaining cores are impacted by the oversubscription actions and the allocation policy. It can be observed that the remaining cores converge to similar points for different safety thresholds in the warm start. We test the C2MARL and the baselines on the warm-start setting for 300 episodes, where the results are shown in Table 2. It can be observed that (1) our C2MARL still outperforms the baselines in different safety-preferences settings. (2) Compared with the cold-start setting (in Table 1), our C2MARL takes the coordination among pre-allocated VMs as well as the oversubscribed VMs into consideration and obtains more saved cores than the cold-start setting. However, the baselines achieve the same saved cores without any improvement which demonstrates the benefits of coordination.

6.7 Azure Cloud
To verify the effectiveness of C2MARL on different patterns of usage rate patterns, we further evaluate our C2MARL in an open public cloud dataset, Azure. The average usage rate of Azure is much higher than the internal Cloud. As shown in Table 3, Grid fails to satisfy the safe constraint due to that most of the users have large usage rates. When taking no oversubscription (i.e. Grid-1.0), the safety constraints are satisfied but no S-Cores then. The MA and SL achieve over-conservative policies that satisfy the safety constraints. Our C2MARL obtains a safety satisfaction over-subscription policy under large usage rate datasets while outperforming others on the S-Cores (i.e. in 0.95 safety threshold, our-0.95 improves more than 26.8% on S-Cores to the MA and SL).

Table 2: Oversubscription in Internal Cloud with Warm Start. The second and third columns are the performance metric with the unit as %. The last three columns are the satisfaction indicator of different levels of safety(α).

| Method | PM-Hot-R | S-Cores | 0.75 | 0.85 | 0.95 |
|--------|----------|---------|------|------|------|
| Grid-0.2 | 95.4 | 80.0 | ✓ | ✓ | ✓ |
| Grid-0.4 | 100.0 | 60.0 | ✓ | ✓ | ✓ |
| Grid-0.6 | 0.0 | 40.0 | ✓ | ✓ | ✓ |
| MA | 0.0 | 49.2 | ✓ | ✓ | ✓ |
| SL | 0.8 ± 1.1 | 49.9 ± 3.3 | ✓ | ✓ | ✓ |
| Our-0.75 | 11.6 ± 10.6 | 69.2 ± 1.5 | ✓ | ✓ | ✓ |
| Our-0.85 | 6.8 ± 6.8 | 65.1 ± 4.8 | ✓ | ✓ | − |
| Our-0.95 | 3.5 ± 1.5 | 69.0 ± 1.4 | ✓ | ✓ | ✓ |

Table 3: Oversubscription in Azure Cloud. The second and third columns are the performance metric with the unit as %. The last three columns are the satisfaction indicator of different levels of safety(α).

| Method | PM-Hot-R | S-Cores | 0.75 | 0.85 | 0.95 |
|--------|----------|---------|------|------|------|
| Grid-0.2 | 100.0 | 80.0 | ✓ | ✓ | ✓ |
| Grid-0.4 | 100.0 | 60.0 | ✓ | ✓ | ✓ |
| Grid-0.6 | 100.0 | 40.0 | ✓ | ✓ | ✓ |
| MA | 0.0 | 8.2 | ✓ | ✓ | ✓ |
| SL | 0.0 ± 0.0 | 7.6 ± 2.2 | ✓ | ✓ | ✓ |
| Our-0.75 | 19.0 ± 3.8 | 15.3 ± 0.13 | ✓ | − | − |
| Our-0.85 | 8.3 ± 3.1 | 14.6 ± 0.1 | ✓ | ✓ | − |
| Our-0.95 | 0.0 ± 0.0 | 10.4 ± 0.2 | ✓ | ✓ | ✓ |

7 CONCLUSION
This paper formulates the oversubscription in Cloud as a chance constraint problem to match the real scene. To solve this problem, we propose an effective chance-constrained multi-agent reinforcement learning method, C2MARL. To reduce the number of chance constraints, C2MARL considers the upper bounds of them and leverages a multi-agent reinforcement learning paradigm to learn a safe and optimal coordination policy. Experimenting on two real-world datasets, C2MARL satisfies all the safety constraints and achieves the largest remaining cores.
A PROOFS

A.1 Proof of Theorem 4

We present the detailed proof of theorem 4 here. For a joint policy \( \pi \), if it satisfies the constraint in Eq. 5 with \( c = (1 - \alpha) \delta \),
\[
\sum_{t=0}^{T} \mathbb{P}(C(t) \mid \sigma_0) / T \leq 1 - \alpha
\]
With the Reverse Markov Inequality, we have
\[
\Pr \left( 1 - \frac{1}{T} \sum_{t=1}^{T} C(t) \leq 1 - \delta \right) \geq \frac{\sum_{t=0}^{T} \mathbb{P}(C(t) \mid \sigma_0)}{T} \geq 1 - \alpha
\]
If
\[
\sum_{t=0}^{T} \mathbb{P}(C(t) \mid \sigma_0) < (1 - \alpha) \delta
\]
then
\[
\Pr \left( 1 - \frac{1}{T} \sum_{t=1}^{T} C(t) \leq 1 - \delta \right) < 1 - \alpha
\]
Due to \( \Pr \left( 1 - \frac{1}{T} \sum_{t=1}^{T} C(t) \leq 1 - \delta \right) = \Pr \left( \frac{1}{T} \sum_{t=1}^{T} C(t) \geq \delta \right) \), and
\[
\Pr \left( \frac{1}{T} \sum_{t=1}^{T} C(t) \geq \delta \right) \geq 1
\]
\[
\Pr \left( 1 - \frac{1}{T} \sum_{t=1}^{T} C(t) < \delta \right) \geq \alpha
\]
The theorem gets proved.

B DISCUSSIONS

This paper proposes an effective method, C2MARL, to learn cooperative oversubscription. This section discusses some key aspects of implementing our methods in the real world.

The trade-off between security and benefits. C2MARL trades off the benefits (saved cores) and security (hotted CPUs) through CPU oversubscription. The hotted CPUs do not cause significant issues in the cloud, as they only occur a few times per week. The subscribers can have different preferences on the benefits and risks. For performance-sensitive subscribers, the over-subscription is not recommended. While CPU over-subscription is recommended for internal clusters that provide resources to non-performance-sensitive internal subscribers. The seldom decreased performance is durable, and a large amount of CPU resources can be saved.
Comparison between SL and C2MARL. The over-subscription problem is a multi-agent sequential decision process, and our C2MARL directly optimizes it. In contrast, SOTA models like SL construct labels for each subscriber based on their usage patterns, which ignores the sequential influence and mutual influences among subscribers and performs worse than C2MARL [25]. Only under strict assumptions (the historical over-subscription and the over-subscription of other subscribers do not influence the optimal over-subscription for ego agent), SL approximates C2MARL. However, the assumptions don’t hold in practice, and C2MARL outperforms SL.

Single-dimension and multi-dimension over-subscription. First, we clarify that our single-dimension over-subscription is practical. In both practice and our simulation, the scheduler handles the main capacity constraints (CPU, Memory, Network, etc.) and ensures these constraints are not being violated. When the CPU is over-subscribed, the hotted CPU on usage is explicitly considered on over-subscription learning and the capacity constraints are handled by the scheduler. Thus the CPU over-subscription does not cause constraints violation on other dimensions.

As for multi-dimension over-subscription, the resources can be oversubscribed only if they can be shared among multiple VMs (e.g., CPU, memory). CPU is one of the most suitable resources for over-subscription as hotted CPU will not cause severe issues with the memory (when the using memory exceeds the node’s memory capacity, the node may be down). Besides, multi-dimension over-subscription poses extra challenges. The multi-dimension over-subscription exponentially increases the action space of over-subscription and further makes the optimization hard.

Over-subscription and using life of resources. The CPU over-subscription does not influence the life of SSDs and DRAMs. These resources are not over-subscribed in our problem and the scheduler ensures safety constraints on them. As for the life of the CPU, we explicitly satisfy the over-hotted CPU status to decrease the influence on the CPU’s life. By setting the safety threshold, we allow the cloud service provider to directly optimize the over-hotted risks. In our experiments, setting 95%-3 safe means that we ensure a node being hot less than 3 times with 95% probability in a week and it does not significantly influence the CPU’s life.

Deployment. The number of physical machines and subscribers can be dynamic when deploying. For the dynamic number of Physical Machines (PMs), the new-added PMs make a change in the cluster status distribution. We suggest training C2MARL with a large c cluster (which has more PMs than the current cluster) and performance is maintained when the practical scenario has fewer PMs. Moreover, the number of PMs in a cluster often does not change a lot otherwise they will be taken as two or more clusters. This further eases the training of C2MARL.

For the dynamic number of subscribers, C2MARL suggests not making over-subscription for newly added subscribers at the beginning. The over-subscription result is highly related to the usage pattern of subscribers and C2MARL (or other learning-based over-subscription methods) should be re-trained when collecting a bunch of usage data. To make optimization more efficient, we suggest training new-added subscribers with the over-subscription rate of others unchanged.

C EXPERIMENT DETAILS

C.1 Hyper Parameters

Table 4: Hyperparameters for C2MARL in internal and Azure cloud. \( a/b \) denote that \( a \) is internal cloud’s parameter and \( b \) is the azure’s.

| Hyperparameter | Value |
|----------------|-------|
| number of agents | 12/9 |
| number of PMs | 500/200 |
| \( \delta \) | 3 |
| \( \beta \) | 0.6 |
| discount factor | 0.9 |
| batch size | 10 |
| memory capacity | 360 |
| target update factor \( r \) | 0.001 |
| Activation function | Relu |
| Optimizer | Adam |

As for the learning rate and dual learning rate, we run a grid search from 1e – 4 to 20 for each setting. For all the baselines, we grid search the hyperparameters for their best performances. All the experiments are trained with requests on 5 weekdays (from Monday to Friday). When it comes to the evaluation, each subscriber’s VM CPU usage in a specific hour-in-a-day (the hour modular 24) is generated from a Gaussian distribution to evaluate the reliability of the obtained policy. The Gaussian distribution’s mean value and standard deviation are determined by the dataset’s statistical mean and standard deviation. Suppose the policy obtains high benefits and satisfies the safety constraints in the stochastic usage distribution. In that case, cloud service providers can be more confident in applying it to the real world.

C.2 Relation between Cluster Hot and PM Hot

C-Hot-R indicates the probability of the cluster having hot PM. Take the Internal Cloud with Cold Start an example. As shown in Table 5, the hot cluster rate is always higher than the PM-Hot-R and our methods meet the desired hot cluster safety.

C.3 Ablation Studies

We make ablation studies on the MARL components of C2MARL. When removing the MARL components of C2MARL, it downgrades to C2MARL w/o MARL [19]. The over-subscription rate of all subscribers is controlled by a single agent and the agent’s action space will grow exponentially as the number of subscribers increases. For Azure, the action space is \( 6^3 \). To make it practical, we allow subscribers to share the same over-subscription rate. In Azure settings with a 95% safety threshold, the C2MARL w/o MARL satisfied the constraints but only resulted in a 2.2% reduction in cores. In contrast, our C2MARL method achieved a 10.4% reduction. These results demonstrate the superiority of our C2MARL method in addressing the over-subscription problem.

Table 5: Oversubscription in Internal Cloud with Cold Start.

| Method | PM-Hot-R | C-Hot-R | S-Cores |
|--------|----------|---------|---------|
| Grid-0.2 | 30.9 | 97.2 | 80 |
| Grid-0.4 | 22.7 | 72.7 | 60.0 |
| Grid-0.6 | 0.0 | 0.0 | 40.0 |
| MA | 0 | 0 | 49.2 |
| SL | 0.2 ± 0.3 | 0.6 ± 0.8 | 49.9 ± 3.3 |
| Our-0.75 | 9.8 ± 0.6 | 21.9 ± 0.3 | 66.1 ± 6.7 |
| Our-0.85 | 6.6 ± 0.6 | 13.9 ± 1.2 | 61.7 ± 9.2 |
| Our-0.95 | 3.5 ± 0.5 | 4.7 ± 0.1 | 59.9 ± 1.9 |