Heterogeneity-aware Task Scheduling based on Personalized Federated Reinforcement Learning

Xin Yong
Xi'an Jiaotong University
Xi'an, Shaanxi, China
xin_yong@stu.xjtu.edu.cn

Li Yan*
Xi'an Jiaotong University
Xi'an, Shaanxi, China
li.yan.88@xjtu.edu.cn

Zhuozhao Li
Southern University of Science and Technology
Shenzhen, Guangdong, China
lizz@sustech.edu.cn

Abstract

The workload data generated in large-scale cloud environments is becoming increasingly complex, making collaborative training a promising approach for developing more efficient task schedulers. Considering privacy security and transfer costs, Federated Reinforcement Learning (FRL) emerges as a promising solution. However, our exploratory experiments demonstrate that the environmental heterogeneity contributes to performance degradation in FRL, which makes the above issue challenging. Accordingly, we propose a Personalized FRL method based on Dual-critic networks and Multi-head attention aggregator (PFRL-DM), which achieves the optimal scheduling policies by collaborative training on diverse workload data in heterogeneous environments without exposing private data. We initially introduce a novel Reinforcement Learning (RL) environment modeling, serving as a foundation for the collaborative training of the cloud scheduling agents. Then, we implement a dual-critic network Proximal Policy Optimization (PPO) algorithm for each client, effectively balancing the influence between global and local models on the agents. Furthermore, we integrate multi-head attention weights into the server-side aggregator to implement personalization. Extensive experiments on various real-world workloads have demonstrated that, compared to state-of-the-art algorithm MFPO, the proposed algorithm exhibits faster convergence, shorter response and completion times, and achieves the highest resource utilization. Additionally, the PFRL-DM algorithm constructs personalized models for each client, enabling greater adaptability in heterogeneous and hybrid workload environments. The codes for this paper can be found at https://github.com/liyan2015/PFRL-DM.

CCS Concepts

• **Theory of computation → Multi-agent reinforcement learning**.

*Corresponding author.

ICPP, San Diego, CA

ACM ISBN 979-8-4007-2074-1/25/09
https://doi.org/3754598.3754602

Keywords

Federated reinforcement learning, task scheduling, proximal policy optimization

ACM Reference Format:
Xin Yong, Li Yan, and Zhuozhao Li. 2025. Heterogeneity-aware Task Scheduling based on Personalized Federated Reinforcement Learning. In *Proceedings of (ICPP)*. ACM, New York, NY, USA, 10 pages. https://doi.org/3754598.3754602

1 Introduction

With the rapid growth of cloud computing, the workloads generated by cloud users are becoming increasingly more dynamic, which may potentially lead to load imbalance, extended response latency, low resource utilization, etc [1, 2]. Driven by these problems, researchers have been focusing on designing efficient resource management and task scheduling approaches for decades [3]. Among them, the DRL-based methods outperform the others due to their capability of learning the optimal decision-making policies via interaction with high dimensional complex task scheduling environments.

With the ongoing migration of enterprise operations to cloud data centers, the continuous emergence of innovative business applications is inducing heightened heterogeneity and complexity in workloads across clouds [4, 5]. However, as cloud providers leveraging traditional task scheduling approaches often rely on isolated workload databases, they cannot single-handedly capture the heterogeneous task characteristics of diverse users for versatile and efficient scheduling of tasks [6, 7]. For instance, banks have attempted to build their own private cloud infrastructures and enhance their operational efficiency [8]. Nevertheless, when banks expect to capitalize on the waves of technology-enabled innovation (e.g., Machine Learning (ML), Artificial Intelligence (AI)), these emerging development demands often fall outside their comfort zone [9]. As a result, the traditional task schedulers trained on isolated datasets are not sufficiently versatile to schedule heterogeneous and dynamic tasks with varying categories, which are rapidly emerging nowadays [6, 10]. This issue is particularly evident in complex scenarios, especially for smaller cloud vendors or private clouds, which lack experience in handling complex tasks that they are not familiar with [11].

To fulfill such emerging demand for scheduling of heterogeneous workloads, it is necessary to establish a versatile task scheduling framework via collaboration across different cloud providers [6, 7]. On the one hand, as shown in Figure 1, traditional approaches are designed for single-cloud environments with isolated workload characteristic data, and hence not qualified to realize this goal. On the other hand, due to high data migration cost and concerns of

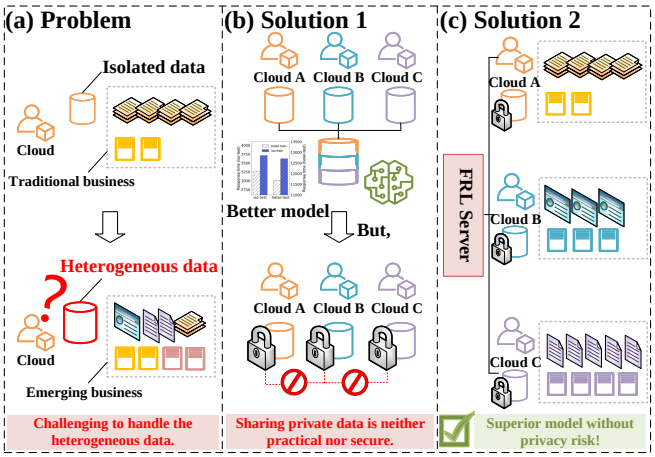

Figure 1: (a) Traditional approaches are not capable of scheduling emerging businesses tasks due to the increasing heterogeneity of the environments. (b) Simply aggregating workload data from different cloud providers is neither practical nor secure. (c) FRL can enable collaborative training of task scheduling models while ensuring data privacy, but have deficiencies.

privacy leakage or trade secrecy, the paradigm of simply aggregating workload data from different cloud providers for the training of a versatile task scheduling model (as advocated in [12, 13]) is not feasible [14].

Federated Reinforcement Learning (FRL), which enables collaborative training of decision-making policies without exposing privacy-sensitive user data [14], is potentially suitable for establishing a versatile task scheduling framework capable of handling heterogeneous tasks and environments without burdensome data migration. However, our preliminary studies in Sections 3.1 and 3.2 demonstrate that the task scheduling environments of different cloud providers are often highly heterogeneous due to their varying service foundations and diverse business involvements. Under such scenarios, existing FRL methods are unable to obtain the optimal decision policy and ensure rapid convergence of model training due to their lack of effective personalization methods that help clients focus on similar models. Although there have been several approaches [15–19] proposed for solving heterogeneity, they have limited capability in personalizing the training strategies, thereby cannot fully adapt to the high heterogeneity of task scheduling environment.

Driven by these limitations and our observations that multi-head attention mechanism is beneficial to improving the performance of task schedulers on respective cloud providers (Section 3.3), we propose a personalized federated parallel task scheduling framework. Specifically, we first conduct a general modeling of the task scheduling problem in cloud computing. Secondly, to address the issue of sub-optimal solutions of global aggregated model, we implement the dual-critic network Proximal Policy Optimization (PPO) algorithm for each client. This mechanism combines public critic networks with local critic networks and adaptively evaluates agents' behavior using optimal weights. Finally, we integrate multi-head attention weights into the server-side aggregator, which determines the optimal attention allocation for each client, subsequently generating a personalized public critic network for each client. In summary, our contributions include:

(1) We utilize multiple cloud workload datasets simulate the different cloud environments. Then we conduct exploratory experiments to investigate the impact of the environmental heterogeneity and the feasibility of generating personalized solutions through multi-head attention mechanism, which motivate the design of our proposed method.

(2) We model the task scheduling problem in the cloud computing environment. To address the environmental heterogeneity in multiple cloud scenarios when applying traditional FRL, we propose a Personalized FRL method based on Dual-critic networks and Multi-head attention aggregator (PFRL-DM).

(3) We implement our framework with PyTorch, and conduct simulation and comparative experiments on real-world workloads. The experiments show that our method can accelerate the learning speed and improve the scheduling performance of agents in heterogeneous environments.

The rest of the paper is organized as follows. Section 2 presents literature review. Section 3 presents the results of our observation experiments. Section 4 gives the details of *PFRL-DM* design. Section 5 presents our experimental studies. Section 6 gives conclusion with future work plan.

2 Related Work

Traditional Task Scheduling Methods for Cloud Computing. Traditional methods generally need numerous iterations to find feasible resource allocation plans, which are unable to optimally allocate tasks in highly dynamic and complex cloud scheduling environments. In contrast, DRL methods are widely embraced due to their flexible adaptability and enhanced autonomous learning capabilities. Chen *et al.* [20] proposed an adaptive and efficient cloud resource allocation scheme based on Actor-Critic DRL to solve the problem of excessive energy consumption and degraded Quality-of-Service (QoS). Li *et al.* [21] proposed a dynamic load balancing scheduling algorithm based on distributional RL to reasonably allocate computing resources for batch jobs. Han *et al.* [22] proposed a DRL-based method to select scheduling algorithms online for edge-cloud jobs, which can capture complex dynamics of Edge-Cloud jobs/tasks. Zhao *et al.* [23] proposed a curiosity-driven collaborative request scheduling scheme in edge-cloud systems, which motivates schedulers to explore the environment both individually and collectively. However, due to the growth of business demands, the isolated task schedulers of the organizations are unable to schedule heterogeneous and dynamic tasks due to their ignorance of unfamiliar tasks [4, 5].

Heterogeneity in FRL. Several methods have been proposed for solving the issues caused by the environmental heterogeneity of FRL. Jin *et al.* [15] studied the constraint of environment heterogeneity defined by different state transitions. Xie *et al.* [16] defined the data heterogeneity types of FRL and proposed a Kullback-Leibler (KL) divergence based penalty to directly constrain the model outputs. Woo *et al.* [17] demonstrated that heterogeneity in federated Q-learning can actually be beneficial when properly leveraged.

Table 1: Machine specifications of different cloud workload datasets.

Cloud Workloads	#CPUs	Mem (GiB)	#Nodes	Platform
Chameleon Cloud [30]	20∼24	7∼62	6	
	48	94∼127	1551	OpenStack
	40	62∼63	101	
CERIT-SC [33]	128	512	20	Kubernetes [31]
	8	64	18	
	8	117	33	Grid-workers [32]
	16	117	113	
	40	232∼488	36	
	40	944∼990	28	
Alibaba [4]	64	512	798	
	96	512	497	
	96	512	280	Alibaba PAI
	96	384	135	
	96	512/384	104	
	96	512	83	

Zhang *et al.* [18] provided an explicit bound on environmental heterogeneity in optimality and validates the benefits of collaboration. The methods FEDSVRPG-M and FEDHAPG-M proposed by Wang *et al.* [19] can exactly converge to a stationary point of the average performance function However, these studies are limited to more controlled environmental heterogeneity, and lack a definition of heterogeneity in highly complex collaborative scheduling environments.

Personalization is an efficient technique for addressing the environmental heterogeneity problem of FL frameworks, which has been largely confirmed in the study of traditional FL techniques [24, 25]. For FRL systems, there is a similar intuition[15], which is demonstrated in many environmental heterogeneity problems dedicated to the study of realistic level [26, 27]. However, these studies focused more on applicability to specific heterogeneous environments, and their personalized approach cannot help clients focus on similar models to obtain the optimal decision strategy.

3 Preliminaries and Motivations

In this section, we present the observations that motivate our system design. The datasets we use for these studies include:

- **Google Dataset** [28] It is collected on a cluster in Google data centers, spanning a period of 29 days in 2011.
- **Alibaba Datasets** [29] They are both collected in Alibaba data centers and released in different years (*Alibaba-2017*,*Alibaba-2018*).
- **HPC Datasets** They are collected from three high performance computing cloud service centers (*HPC-KS*, *HPC-HF* and *HPC-WZ*) for academic research.
- **Chameleon Cloud Datasets** [30]. They are collected from two OpenStack cloud centers (*KVM-2019* and *KVM-2020*) operated by the Chameleon testbed for educational projects.
- **CERIT-SC Workloads**. They are collected from two CERIT Scientific Cloud Centers (*K8S* [31] , *CERIT-SC* [32]).

Table 2: Environmental settings for different clients.

	Machine specifications (CPU, Memory, Counts)	Workload Dataset
Client 1	(16,128,4) (32,256,1)	*Google*
Client 2	(32,256,3)	*Alibaba-2017*
Client 3	(16,128,2) (32,256,2)	*HPC-HF*
Client 4	(16,128,3) (32,256,2)	*KVM-2019*

3.1 Heterogeneity of Cloud Task Scheduling Environments

As mentioned before, the workload datasets are derived from real-world execution logs collected across varying time periods and machine specifications. Figure 2 and 3 illustrate a comprehensive distribution of the requested CPU and memory usage within the workload datasets. Figure 4 and 5 illustrates the measured hourly task arrival rates and the Cumulative Distribution Function (CDF) of task execution time among different workload datasets, respectively. It is evident that the resource demands, arrival time and execution period of most applications in diverse cloud workloads exhibit significant heterogeneity.

Furthermore, Table 1 illustrates the specification details of machines in several cloud workload datasets. Apparently, the machine specifications exhibit significant variation across different clusters. Considering the factors mentioned above related to state transition probabilities, the Markov Decision Processes also vary significantly across different scheduling systems.

To further investigate and analyze the challenges posed by such heterogeneity to task scheduling in traditional cloud environments, we conducted a series of exploratory experiments. Specifically, to model the machine specification heterogeneity illustrated in Table 2, we establish an OpenAI Gym environment that considers various resource types (e.g., CPU, memory). Figure 6 illustrates a toy example of state representation in the established environment, where the *Remaining Capacity* indicates the available resources in the VMs, the *Running State of vCPUs* reflects the current vCPU condition, and the *Waiting Queue Tasks* shows the information (including requested vCPUs and memory) of tasks in the waiting queue. The figure depicts the current state at time step 6 in a cloud environment with three VMs of varying resource capacities. Only one vCPU is available in the first VM, while the other two VMs are fully occupied by tasks. The first task in the current waiting queue, represented as (2, 20), requires 2 vCPUs and 20 GB of memory. Obviously, this task cannot be executed immediately and must wait for the completion of tasks on either VM 1 or VM 3 before it can be scheduled. More details about the RL environment modeling of task scheduling problem are given in Section 4.1.

In our experiment, 4 cloud scheduling environments are constructed by diverse workload datasets listed in Table 2. To effectively capture the essential patterns of heterogeneous environments, we represent task characteristics as an isolated distribution and generate 3500 samples from each dataset. For each environment, 60% tasks are used for training (denoted as *iso-train*), the rest 40% tasks are used for testing (denoted as *iso-test*). Similarly, we combine these extracted tasks to build a heterogeneous cloud task dataset, of which 60% tasks are used for training (denoted as *heter-train*), and the rest 40% tasks are used for testing (denoted as *heter-test*). The local PPO models are trained using the Adam optimizer with

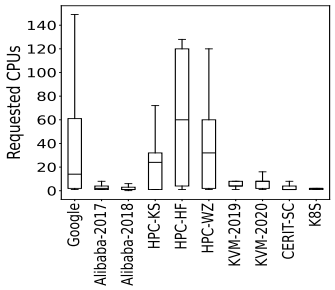

Figure 2: Distribution of task requested CPU usages.

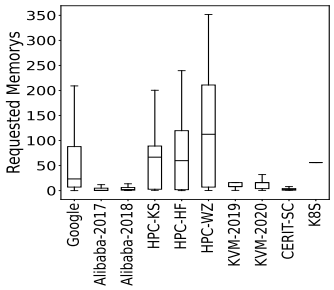

Figure 3: Distribution of task requested memory usages.

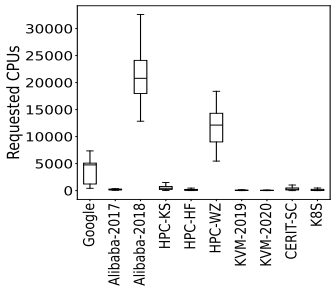

Figure 4: Distribution of task arrival rates

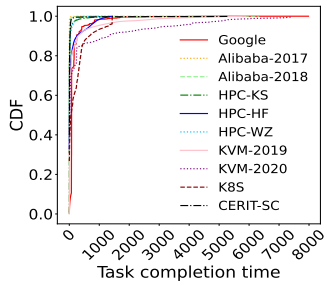

Figure 5: CDF of task completion time.

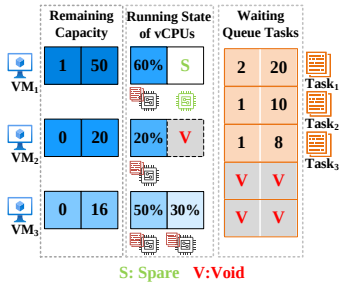

Figure 6: An example of state representation in the proposed RL environment.

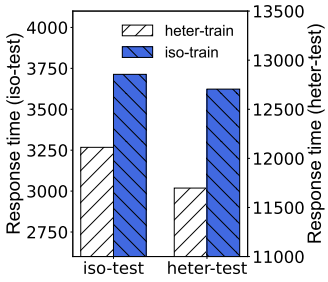

Figure 7: Average response time on test datasets of isolated and combined training.

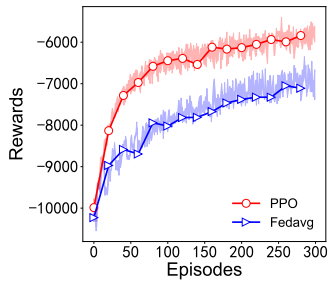

Figure 8: Traditional FRL underperforms compared to the independent PPO algorithm.

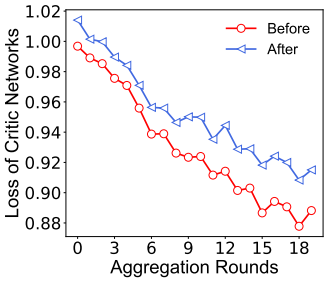

Figure 9: Aggregated critic networks incur higher losses than local models.

the actor learning rate of 3×10^{-4} and the critic learning rate of 1×10^{-4}. Each PPO agent employs a neural network with a single hidden layer consisting of 64 neurons. In addition, the discount factor is 0.99, the clip parameter is 0.2 and the training episodes is set as 300. We evaluate the performance of the PPO-based scheduler with different combinations of training datasets and testing datasets. Specifically, the scheduler is trained with *iso-train* and *heter-train* per environment, respectively. The trained scheduler is tested with *iso-test* and *heter-test* per environment, respectively. Finally, the response time of the tasks are collected under different combinations of training datasets and testing datasets.

Figure 7 illustrates the measured results of *iso-train* and *heter-train* on *iso-test* and *heter-test*. We can see that the RL models trained on diverse cloud workloads achieve lower average response time on all the test environments, demonstrating that the combined heterogeneous workload dataset is more conducive to generating superior schedulers than the isolated datasets. However, for isolated users, sharing workload datasets with others is challenging due to concerns about privacy risk and data security. This underscores the necessity for cloud providers to adopt FRL for collaborative training, which enables the development of more effective scheduling models.

3.2 Challenges of Environmental Heterogeneity in FRL

Driven by the observations above, we tentatively apply traditional FRL on the collaborative training of the schedulers with heterogeneous scheduling environments. Traditional FRL typically employs a centralized server and multiple clients, with the primary objective of maintaining an optimized global model [15]. However, the complexity and heterogeneity of the scheduling environments, as outlined in Section 3.1, restrict the performance of FRL techniques that rely on averaging strategies. To further investigate this assumption, we conduct exploratory experiments of FRL.

For the sake of brevity, we utilizes the *iso-train* environments mentioned before as the scheduling environments for the 4 clients, with an FRL aggregation server configured accordingly. Then the FedAvg technique is applied to these clients, where the total number of episodes is 300, and the communication frequency is 15 episodes. For comparison, the agents are also trained independently with the same configuration of PPOs. Figure 8 illustrates the comparison of performance curves between the FedAvg and independent PPO algorithms. It is evident that the Fedavg algorithm has a lower convergence speed, which is speculated to be affected by the heterogeneous environments in the global aggregation process. Furthermore, considering that the evaluation performance of the critic network is crucial to the policy update direction of the PPO algorithm, we record the average loss generated by critic networks of local agents before and after each aggregation, as illustrated in Figure 9. It can be seen from the figure that the agents perform worse in evaluating the value of actions after the global aggregated model is loaded in agents. In summary, environmental heterogeneity contributes to the poor performance of the FedAvg method. More specifically, this heterogeneity directly impacts the agents' networks, e.g., the aggregation models reducing the evaluation performance of the local agent's critic network.

For each client, environmental heterogeneity constrains its capability for improvements, hindering the federation mechanism from achieving its full potential. Therefore, if we can effectively balance and leverage both the client's local experience and the global aggregated policies by combining the performance of the two critic

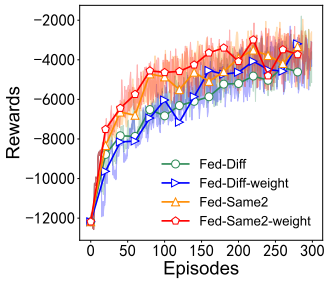

Figure 10: For a client, focusing more on similar clients accelerates convergence.

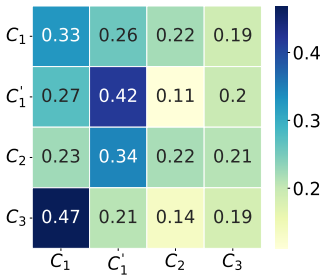

Figure 11: Multi-head attention assigns more weights to similar models.

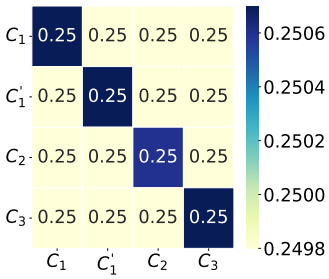

Figure 12: KL-divergence can't generate weights focusing on similar clients.

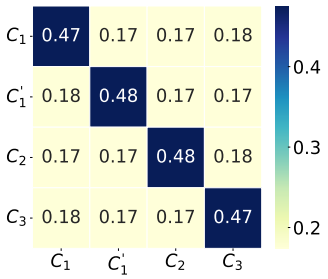

Figure 13: The cosine similarity weights fails to focus on similar clients.

networks, it could help mitigate the suboptimal solution issue in the training.

3.3 Challenges in Generating Personalization Models

Addressing environmental heterogeneity is the key challenge for applying FRL to collaborative task scheduling scenarios. In retrospect, numerous studies have concentrated on the heterogeneity issue of federated mechanisms. Jiang et al. [34] have suggested that personalized methods should be developed to manage such heterogeneity within the FL framework. Mansour et al. [35] argued that aggregating clients with more similarities helps in personalization. Therefore, we investigate the feasibility of leveraging similarities of agents in FRL for such improvements.

We adopt the same environment setups with 4 isolated agents as described earlier, while employing the following different combinations and weight assignments for comparative experiments: a) *Fed-Diff*: 4 different clients C_1, C_2, C_3, C_4 with averaging strategies; b) *Fed-Diff-weight*: 4 different clients C_1, C_2, C_3, C_4 while C_1 pays more weights to C_2; c) *Fed-Same2*: 2 same clients $C_1, C_1^{'}$ and 2 different client C_3 and C_4 with averaging strategies; d) *Fed-Same2-weight*: 2 same clients $C_1, C_1^{'}$ and 2 different client C_3 and C_4, while C_1 pays more weights to $C_1^{'}$. The convergence curves of the client C_1 of these Fedavg configurations are illustrated in Figure 10. It can be found that if clients pay more attention on similar clients, they can achieve more improvements.

The multi-head attention mechanism facilitates the automatic focus on deeper similarity information. We compare the weights generated by common similarity measures (KL divergence, cosine similarity) with the multi-head attention mechanism. To be more specific, we set up 4 clients $C_1, C_1^{'}, C_2, C_3$. Client C_1 and client $C_1^{'}$ are trained in the same environments. The input to the similarity-based weight generation methods consists of the critic models of these clients, and the output weights are visualized in the heatmaps shown in Figure 11 to Figure 13. As is shown in the Figure 11, the heatmap of the multi-head attention mechanism indicates that Clients C_1 and $C_1^{'}$ assign greater attention to each other compared to other clients, demonstrating the availability of the attention mechanism. However, the KL divergence and cosine similarity fail to work. We suspect that the multi-head attention mechanism may capture richer information rather than being limited to distribution features and spatial locations in this condition.

Inspired by the above exploratory experiments, we focus on developing a personalization method based on the multi-head attention mechanism.

3.4 Threat Model

Security threats within the FL framework have raised significant concern from researchers in recent years [36]. In this article, the FL server is assumed as "honest-but-curious" [37]. We also assume that the clients will employ advanced techniques to safeguard their private information against attacks. The protection of FL framework [38, 39] against other forms of malicious activities falls outside the primary focus of this work.

4 System Design

4.1 Problem Statement

We consider a federated framework involving N clouds. For each client, the scheduling agent typically interact in completely independent environments. It is important to note that, by definition of FRL [40], the clients are expected to have similar definitions of the RL environments.

Based on this, we model the scheduling environment for each cloud (Section 4.2). We assume that the maximum number of VMs within the clusters is L. Each VM comprises d resource types(e.g., vCPU, memory), with the upper limit of each resource type being denoted by U (e.g., U^{vcpu}, U^{mem}). Similar to prior work, the resource demands of tasks are assumed to be known to the system upon arrival.

4.2 RL formulation

State Space: The state space represents the set of states that an agent can explore. We summarize the previous studies on system state modeling into two approaches: statistical values [20, 41] and image-based representations [2, 42]. Considering scalability and simplicity, we integrate these two methods to model the state space. As shown in the Figure 6, the defined state space consists of 3 main parts: *Remaining Capacity*: S^{VM}, the *Running State of vCPUs*: S^{vCPU}, and the *Waiting Queue Tasks*: S^{Queue}.

$$S = (S^{VM}, S^{vCPU}, S^{Queue}) \quad (1)$$

where the remaining capacity of the VMs in this cloud $S^{VM} = \{(m_i^1, ...m_i^d), i \leq L\}$. Therefore, the running state $S^{vCPU} = \{O_i^k, i \leq L, k \leq U^{vcpu}\}$ is used to describe the state of the vCPUs running the tasks. O_i^k represents the occupied state of vCPU k in VM i, i.e., the percentage of completed progress for the running task

on this vCPU. S^{Queue} shows information about the tasks that are currently waiting in the queue. $S^{Queue} = \{(j_i^1, ...j_i^d), i \leq Q\}$ shows the requested resources of the task i, and Q represents the length of the waiting queue.

It is worth noting that we won't directly include the running time/length of the task in the observation like the prior works [2, 20, 41, 42]. Instead, we assume that the VM could track the task's completion progress, enabling the agent to autonomously perceive task information. This approach enhances the realism of the simulation and modeling, making it more consistent with actual conditions. For clouds with an insufficient number or capacity of VMs (e.g. the positions marked as *void* in Figure 6), it is necessary to pad the corresponding positions.

Action Space: For each task, the scheduler will assign a VM to complete it. Therefore, the action space is:

$$A = (1, ...L, -1) \tag{2}$$

where -1 represents that there is no VM selected for this action and the scheduler will do nothing in this step.

Reward function:

The reward function serves as an estimate of the rewards that an agent gains by executing specific actions within various states. In this paper, the reward function is formulated aiming at decreasing the response time and improving the load balancing, which is the focus of most cloud providers. The response time is defined in Equation (3), where $\mathrm{j}_i^{\mathrm{run}}$ and $\mathrm{j}_i^{\mathrm{wait}}$ represent the execution time and waiting time of the task i.

$$\mathrm{j}_i^{\mathrm{res}} = \mathrm{j}_i^{\mathrm{run}} + \mathrm{j}_i^{\mathrm{wait}} \tag{3}$$

We then define load balancing [21] $LoadBal(t)$ to measure the degree of load balancing at current time t.

$$LoadBal(t) = \sum_{i=1}^{d} w_i \sqrt{\frac{\sum_{m \in M_n} (m^{load}(t,i) - AvgLoad(t,i))^2}{|M_n|}} \tag{4}$$

where w_i weights the importance of the resource type i. Load $m^{load}(t,i) \in [0,1]$ is defined by the ratio of remains of the resource type i on VM m at time slot t to the total amount of it. $AvgLoad(t,i)$ represents the average load of resource type i among the VMs, which is defined in Equation (5). M_n represents the VM collection of the nth client and $|M_n|$ is the number of VMs in it.

$$AvgLoad(t,i) = \frac{\sum_{m \in M_n} m^{load}(t,i)}{|M_n|} \tag{5}$$

In order to enable the agent to learn a strategy that not only satisfies the Quality of Service (QoS) requirements by decreasing response time but also achieves load balancing among VMs within the system as much as possible, we define the reward value as follows:

$$\mathcal{R}_a = \rho R_{res} + (1-\rho) R_{Load} \tag{6}$$

where R_{res} and R_{Load} represent the rewards related to response time and load balancing, respectively. ρ is used to weight the rewards. R_{res} and R_{Load} are defined as follows:

$$R_{res} = e^{\mathrm{j}_i^{\mathrm{run}}/\mathrm{j}_i^{\mathrm{res}}} \tag{7}$$

$$R_{load} = \begin{cases} 1, \text{if } Load_c \leq 0 \\ Load_c, \text{else} \end{cases} \tag{8}$$

In Equation (7), j_i^{run} is utilized for normalization. As the response time increases, the corresponding reward value diminishes accordingly. In Equation (8), $Load_c = LoadBal(t^{'}) - LoadBal(t)$, it represents the impact degree on load balancing before and after the deployment of the task. The lower value of $LoadBal$ indicates a more balanced load distribution. In other words, when the task deployment contributes to a more balanced load ($Load_c \leq 0$), a higher reward value will be attained. In order to mitigate the sparsity of rewards acquired by the agent, certain behaviors under specific scenarios are penalized. If a scheduled task requires more resources than the VM can provide, the action will be denied and penalized based on the utilization of the VM in Equation (9).

$$\mathcal{R}_p = -e^{\sum_{i=0}^{d} w_i m_a^{\mathrm{uti}}(t,i)} \tag{9}$$

where $m_a^{uti}(t,i)$ represents the utilization of resource i for the VM a at the current time t. Moreover, if the scheduler tends to wait for the next time step due to inertia policies despite the presence of VMs that meet the requisite conditions, the reward will be a larger negative constant. In addition, the reward function can be easily extended to accommodate other optimization objectives, such as makespan, cost, energy consumption and so on.

4.3 Local Dual-Critic Network PPO

Drawing on the exploration detailed in Section 3.2, we aim to leverage the enhancement potential of the global model while mitigating the adverse effects of suboptimal solutions that arise from heterogeneity. As highlighted by Mansour et al. [35], personalized models can be regarded as intermediate models, bridging the gap between purely local and global models. In this article, we investigate the combination of global and local models to explore the feasibility of achieving personalization through it.

Inspired by the research of Marfoq et al. [43] on model interpolation methods, we incorporate both a global model and a local model for each client to preserve certain local information. The proposed method employs two critics for each client: a public critic and a local critic. The public critic represents the global model received by the client from the server during the training process, whereas the local critic denotes the client's critic model. The client-server interaction is detailed in Section 4.5. In this section, we primarily focus on the training process of the local clients. The two critic networks collaboratively evaluate the states, and the evaluated values are determined by their weighted results. During the update process, they will also be optimized synchronously. Suppose the policy network of PPO trained by the nth client can be represented as π_{θ_n} and its parameters are θ_n. The parameters of the local value network and public value network are defined as ϕ_n and ψ_n, respectively. Then the objective function of it can be formulated as follows:

$$\theta_n^{k+1} = \underset{\theta_n}{argmax} \underset{s_n, a_n \sim \pi_{\theta_n^k}}{E} [J(s_n, a_n, \theta_n^k, \theta_n)] \tag{10}$$

where E represents the mathematical expectation and J is shown in Equation (11): .

$$J(s_n, a_n, \theta_n^k, \theta_n) = \min \Big(\frac{\pi_{\theta_n}(a_n|s_n)}{\pi_{\theta_n^k}(a_n|s_n)} A^{\pi_{\theta_n^k}}(s_n, a_n), \; g(\epsilon, A^{\pi_{\theta_n^k}}(s_n, a_n)) \Big) \tag{11}$$

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A \leqslant 0 \end{cases} \tag{12}$$

where $\pi_{\theta_n^k}$ is the old parameterized policy compared to the new policy π_{θ_n}. ϵ is a small hyperparameter.

As for the advantage function $A^{\pi_{\theta_n^k}}$, it can be defined as:

$$A^{\pi_{\theta_n^k}}(s_n, a_n) = Q^{\pi_{\theta_n^k}}(s_n, a_n) - V^{\pi_{\theta_n^k}}(s_n) \tag{13}$$

where $Q^{\pi_{\theta_n^k}}(s_n, a_n)$ is the action-value function estimates by samples, and $V^{\pi_{\theta_n^k}}$ represents the approximation of the state-value function.

$$V^{\pi_{\theta_n^k}}(s_n) = \alpha V_{\phi_n^k}(s_n(t)) + (1-\alpha) V_{\psi_n^k}(s_n(t)) \tag{14}$$

where $\alpha \in [0,1]$ represents the weight of the local critic network. Each time the model parameters change, including the local update and receiving the global model, the two critic networks will be evaluated according to the trajectories in the buffer, and the losses obtained are denoted as $L^{\phi_n^k}$ and $L^{\psi_n^k}$. Then, the weight parameter α is adaptively updated as shown below:

$$\alpha = \frac{e^{-L^{\phi_n^k}}}{e^{-L^{\phi_n^k}} + e^{-L^{\psi_n^k}}} \tag{15}$$

The local and global value functions also need to be updated via gradient descent algorithm.

$$\phi_n^{k+1} = \arg\min_{\phi_n} \frac{1}{|D_n^k|T} \sum_{\tau \in D_n^k} \sum_{t=0}^{T} \Big[V_{\phi_n^k}(s_n(t)) - r_n(s_n(t), a_n(t)) \Big]^2 \tag{16}$$

$$\psi_n^{k+1} = \arg\min_{\psi_n} \frac{1}{|D_n^k|T} \sum_{\tau \in D_n^k} \sum_{t=0}^{T} \Big[V_{\psi_n^k}(s_n(t)) - r_n(s_n(t), a_n(t)) \Big]^2 \tag{17}$$

4.4 Multi-head Attention Aggregator

Attention mechanism enhances the focus on correlations between data features. Consequently, it is also considered promising in the field of researching personalization in FL and FRL [44]. The preliminary experiments in Section 3 indicates that the multi-head attention mechanism contributes to explore the similarities between clients. Therefore, we introduce the multi-head attention mechanism into the aggregation process on the server side in FRL for personalization. We adopt a similar approach to attention weight generation as described in Section 3.3. Extending this approach, attention weights are utilized to create personalized aggregation models tailored to each client's focus, rather than relying on a generalized average model. The formula for the attention is presented in Equation (18).

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{18}$$

Traditionally, the attention mechanism is replicated by concatenation to obtain multi-head attention as

$$\text{MultiHead} = (Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{19}$$

where head_i is defined as follows.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{20}$$

4.5 Overview Structure

The overall structure of the proposed method is shown in Figure 14. Its pseudo-code is shown in Algorithm 1. As illustrated in Figure 14, our proposed algorithm is mainly divided into three stages. In the first stage, each agent mainly utilizes the local dual-critic network PPO algorithm for training. For the next stage, considering the communication delay and computational complexity, the server will perform personalized aggregation based on the multi-head attention mechanism once it receives the trained public critic network parameters from K clients. Finally, the clients involved in the aggregation will receive their personalized models, while

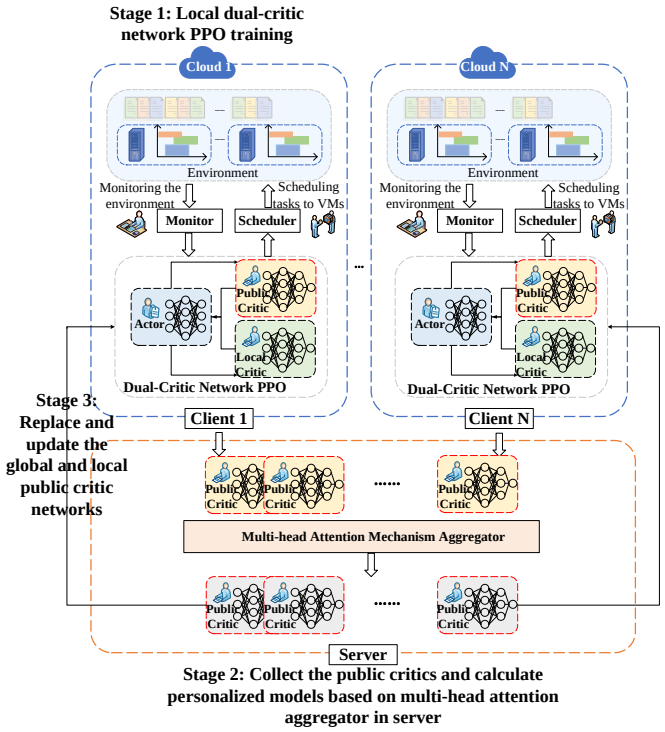

Figure 14: System model and learning framework of PFRL-DM.

the others will obtain the model consistent with the one stored on the server. Then the clients reset the local critic network and the public critic network for another round of training. This approach enables each agent to maintain a balance between local updates and global personalized models received from the server for achieving a satisfactory model even in the environments with heterogeneity.

5 Performance Evaluation

5.1 Experimental Settings

Comparison Methods. To the best of our knowledge, there is no FRL research on heterogeneous cloud workload for the task scheduling problem. Therefore, we compare the proposed FRL algorithm with the standard Fedavg framework [45], state-of-the-art FRL method MFPO [46] and independent PPO [47], which serves as the baseline. The PPO-based algorithms utilize the same parameter settings as those described in Section 3.1 and $K = \frac{N}{2}$ for the proposed method. The parameters of MFPO are kept consistent with the original article throughout the experiment.

Datasets. We conduct task scheduling simulation experiments using the datasets presented in Section 3. Specifically, we consider the workload datasets as distributions and sample 3500 tasks for each client. As for the capacity and number of VMs owned by each cloud provider, we randomly set them by referring to the machine specifications defined by the cloud workloads as well as common VM configuration on the market. As shown in Table 3, the columns represent the machine specifications and datasets of diverse clients, respectively. To better simulate the task scheduling environments

Algorithm 1: PFRL-DM algorithm

Input: Initial public value function parameters $\psi_G^{(0)}$ of server and number of local update epochs Ω

for *each communication round* $m = 0, 1, 2, \ldots$ **do**
  {Local training at clients}
  for *each client* n *in parallel* **do**
    if $m = 1$ **then**
      Replace both local critic network $\phi_n^{(0)}$ and public critic network $\psi_n^{(0)}$ with $\psi_G^{(0)}$;
    else
      Replace local critic network $\phi_n^{(m)}$ with $\psi_n^{(m)}$;
      Replace public critic network $\psi_n^{(m)}$ with $\psi_n^{(m+1)}$;
      Update parameter α by Equation(15)
    Train in local environment for Ω epochs;
  {Global operations at the server}
  The server collects $K (K \leq N)$ local public critic network models $\{\psi_k^{(m)}\}_{k=1}^K$
  The server generates the multi-head attention weights $W^{(m)} = (w_1, \ldots w_K)$ in calculating by Equation (18);
  The server aggregates and calculates the personalized models:

$$\psi_k^{(m+1)} = w_k \cdot \Theta^{(m)} \tag{21}$$

  where $\Theta^{(m)}$ is a matrix of all the $\psi_k^{(m)}$ collected from the clients;
  The server updates the global public critic network by Equation (22), which also serves as the update model for the uninvolved clients:

$$\psi_G^{(m+1)} = \frac{1}{K} \sum_{k=1}^{K} \psi_j^{(m+1)} \tag{22}$$

of different entities, the above settings incorporate the diversity and heterogeneity of workload datasets and machine configurations.

Metrics. To observe the convergence behavior of the proposed method, we measure the total reward during the FRL training. To analyze the generalization abilities, the following metrics are also used to measure improvements of the cloud schedulers.

Response time: The response time measures the mean duration from the arrival of tasks to their completion. This metric is closely associated with QoS requirements [20], where a lower response time generally indicates a more effective scheduling strategy and a higher quality of user experience [1].

$$Avg\ Response = \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} j_i^{res} \tag{23}$$

where $\mathcal{G}$ represents the collection of the tasks and $|\mathcal{G}|$ is the number of tasks in it.

Makespan: Makespan typically measures the time span from the start of scheduling to the completion of the last task in a sequence, providing a macro-level and global vision of the scheduler's performance. Minimizing the makespan is generally a objective the system aim to achieve.

Utilization: It is widely acknowledged that an inefficient scheduler may lead to excessive resource wastage. Therefore, resource utilization serves as a valuable metric to assess whether a scheduling strategy is sufficiently effective and satisfactory [2].

$$Avg\ Util = \sum_{i=0}^{d} \frac{w_i \sum_{m \in M_n} \sum_{t=0}^{T} m^{uti}(t,i)}{|M_n| T} \tag{24}$$

where $m^{uti}(t,i)$ represents the utilization of the VM m on resource i in time step t.

Load balancing: The load balancing is defined by Equation (4). For clarity, the metric definition of average load balancing is given here.

$$Avg\ LoadBal = \frac{\sum_{t=0}^{T} LoadBal(n,t)}{T} \tag{25}$$

where $LoadBal(t)$ represents the load balancing of the scheduling system at current time t.

Implementation. We conduct evaluations on a Linux server with 1 A100 GPU, 1 Intel Xeon CPU, and 256 GB memory.

Table 3: Environmental settings for different clients.

	Machine specifications (CPU, Memory, Counts)	Workload Dataset
Client 1	(8,64,1) (16,128,4) (64,512,2)	*Google*
Client 2	(8,64,3) (32,128,3) (64,512,1)	*Alibaba-2017*
Client 3	(8,64,3) (32,256,2) (64,512,2)	*Alibaba-2018*
Client 4	(8,64,2) (32,256,3) (40,256,2)	*HPC-KS*
Client 5	(8,64,1) (48,256,2) (64,512,3)	*HPC-HF*
Client 6	(16,128,1) (32,256,3) (40,256,3)	*HPC-WZ*
Client 7	(16,128,1) (40,256,3) (32,200,3)	*KVM-2019*
Client 8	(16,128,4) (64,512,1)	*KVM-2020*
Client 9	(8,64,2) (16,128,2) (64,512,1)	*CERIT-SC*
Client 10	(8,128,2) (16,128,4)	*K8S*

5.2 Training Performance Analysis

We first compare the training process of different FRL algorithms when training episodes is 500 and communication frequency is 25 episodes, which is shown in Figure 15. Compared to the independent PPOs, the proposed PFRL-DM converges faster under the collaborative training framework, which indicates that the collaborative efforts of multiple agents can facilitate early exploration. Furthermore, by analyzing the FedAvg curve, it is evident that FL framework yields sub-optimal solutions due to the environmental heterogeneity, hindering the training progress toward optimal results. These facts demonstrate that the PFRL-DM effectively addresses the challenges posed by environmental heterogeneity, leveraging co-training to enhance both the convergence rate and the performance of clients within their own specific environments. In addition, the PFRL-DM algorithm requires only the transmission costs of the public critic networks, whereas traditional FedAvg necessitates both the actor and critic networks.

The proposed PFRL-DM algorithm is also more advantageous compared to the state-of-the-art research MFPO. The performance of the MFPO algorithm improves steadily but fails to reach an optimal level, which we speculate is due to its inability to account for environmental heterogeneity, while its momentum mechanism preserves the influence of past solutions. This observation underscores the need for FRL research to address the heterogeneity of its

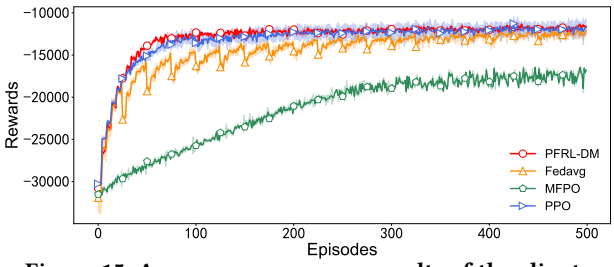

Figure 15: Average convergence results of the clients.

application environment, as neglecting this factor may impede the achievement of the desired improvements.

5.3 Generalization Performance Analysis

This section presents two experiments to evaluate the generalization performance of the model generated during the training phase. Firstly, we test the local models on hybrid workloads to simulate the complex and dynamic real-world scenarios. To be more specific, we assume that VM specifications within a single cloud provider are generally stable and seldom changed, while workload data is diverse and subject to frequent changes. Accordingly, we generate the workload test set on each client by the following steps: 20% of the original dataset is retained, while the remaining portion is randomly drawn from the datasets of the other 9 clients. Then, we test the performance of each client's policy model trained in the training phase in the new complex environments with same VMs but hybrid workloads. Figures 16-19 present the distribution of the average results among the participated clients. It can be found that the proposed model performs better than other comparison algorithms on the test sets of most clients, which proves the effectiveness of the algorithm. From a user perspective, reduced response time and optimized makespan significantly enhance satisfaction levels. For cloud providers, improved resource utilization and enhanced load balancing represent critical operational advantages.

For clarity, we also applied the pair-wise Wilcoxon Signed-Rank test between PFRL-DM and each of the other methods, of which test results (p-values) are shown in Table 4. We can see that all the p-values are less than 0.05, indicating that there are statistically significant difference between PFRL-DM and the other methods (i.e., PFRL-DM outperforms the others).

Then, we test the performance of the proposed algorithm when adding a new agent, simulating real-world scenarios in which a new client is integrated into the federation. As shown in Figure 20, the new agent with the same environment settings as the Client 1 joins in the 100th episodes. The figure illustrates the curve of the new scheduler gradually converging in PFRL-DM aggregation and the curve of training a new PPO scheduler in this environment. At time step 0, a new agent is introduced. PFRL-DM initializes this agent with the model provided by the server, whereas PPO relies on a randomly generated one. The fact that the former immediately earns higher rewards demonstrates that the aggregation-based initialization is significantly more effective than randomization. The faster convergence also indicates that the personalized mechanism enables new agents to more rapidly perceive the environment.

Table 4: The pair-wise Wilcoxon Signed-Rank test results (p-values) between PFRL-DM and the others.

Metrics	Fedavg	MFPO	PPO
Average response	1.93×10^{-3}	1.93×10^{-3}	1.93×10^{-3}
Average makespan	1.93×10^{-3}	1.93×10^{-3}	1.93×10^{-3}
Average resource utilization	1.93×10^{-3}	1.93×10^{-3}	1.93×10^{-3}
Average load balancing	1.93×10^{-3}	1.93×10^{-3}	1.93×10^{-3}

5.4 Effect of Communication Frequency

Communication frequency typically refers to the number of iterations of the local model between two communications. Various communication frequencies are selected for testing, and the results are depicted in Figure 21. The figure demonstrates that communication frequency does indeed affect the algorithm's convergence performance; however, the differences are generally not substantial.

6 Conclusion

In this paper, we have proposed a personalized FRL method named PFRL-DM. This method employs a PPO technique based on a dual-critic network architecture on the client side. On the server side, a specific aggregator based on multi-head attention weights is designed to generate the personalized models. Experimental results on 10 real-world cloud workload datasets demonstrate that the PFRL-DM algorithm achieves faster convergence and outperforms the compared methods in adapting to heterogeneous and hybrid workload environments. For future work, we plan to further explore the application of the proposed algorithm on workflow datasets with dependencies and other scheduling problems.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant No. 2022YFB4500800), the National Natural Science Foundation of China (Grants No. 62103325 and 62202216), the Shaanxi High-Level Talent Program (Grant No. 2021QCYRC4-26), the Guangdong Basic and Applied Basic Research Foundation Grant (No. 2023A1515010244), and the Shenzhen Science and Technology Program (Grant No. 20231121101752002).

References

[1] Jialin Lu, Jing Yang, Shaobo Li, Yijun Li, Wu Jiang, Jiangtian Dai, and Jianjun Hu. 2024. A2C-DRL: Dynamic Scheduling for Stochastic Edge–Cloud Environments Using A2C and Deep Reinforcement Learning. *IEEE IOT* 11, 9 (2024).

[2] Shanka Subhra Mondal, Nikhil Sheoran, and Subrata Mitra. 2021. Scheduling of Time-Varying Workloads Using Reinforcement Learning. In *Proc. of AAAI*.

[3] Mohit Kumar, S.C. Sharma, Anubhav Goel, and S.P. Singh. 2019. A comprehensive survey for scheduling techniques in cloud computing. *JNCA* 143 (2019).

[4] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 2022. MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. In *Proc. of NSDI*.

[5] Zhisheng Ye, Wei Gao, Qinghao Hu, Peng Sun, Xiaolin Wang, Yingwei Luo, Tianwei Zhang, and Yonggang Wen. 2024. Deep Learning Workload Scheduling in GPU Datacenters: A Survey. *ACM Comput. Surv.* 56, 6 (2024).

[6] Zongheng Yang, Zhanghao Wu, Michael Luo, Wei-Lin Chiang, Romil Bhardwaj, Woosuk Kwon, Siyuan Zhuang, Frank Sifei Luan, Gautam Mittal, Scott Shenker, et al. 2023. {SkyPilot}: An intercloud broker for sky computing. In *Proc. of NSDI*.

[7] Ion Stoica and Scott Shenker. 2021. From Cloud Computing to Sky Computing. In *Proc. of HotOS*.

[8] 2024. Banking as a Service (BaaS) Platform Market. https://www.futuremarketinsights.com/reports/banking-as-a-service-baas-platform-market. Accessed in December, 2024.

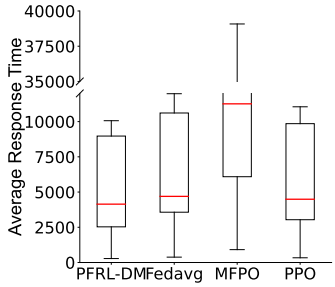

Figure 16: Average response time of the test datasets.

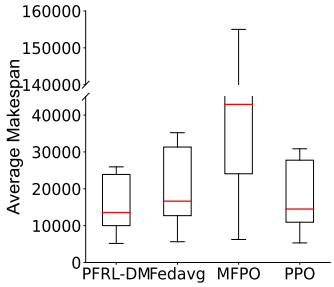

Figure 17: Average makespan of the test datasets.

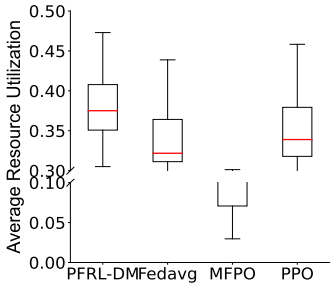

Figure 18: Average resource utilization of the test datasets.

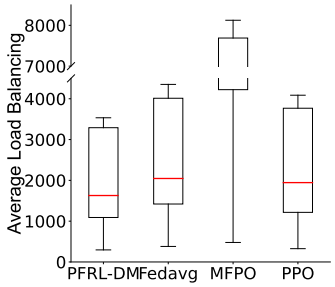

Figure 19: Average load balancing of the test datasets.

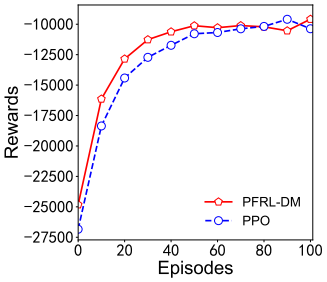

Figure 20: Generalization results of adding a new agent.

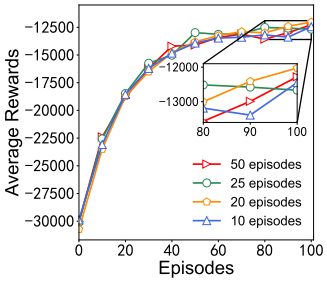

Figure 21: Impact of communication frequency.

[9] 2024. Changing the game: the impact of artificial intelligence on the banking and capital markets sector. www.deloitte.com/global/en/Industries/financial-services/perspectives/changing-the-game.html. Accessed in December, 2024.

[10] Wencong Xiao, Romil Bhardwaj, Ramachandran Ramjee, Muthian Sivathanu, Nipun Kwatra, Zhenhua Han, Pratyush Patel, Xuan Peng, Hanyu Zhao, Quanlu Zhang, Fan Yang, and Lidong Zhou. 2018. Gandiva: Introspective Cluster Scheduling for Deep Learning. In *Proc. of OSDI*.

[11] Francisco Brasileiro and Eduardo Falcão. 2016. Federation of Private IaaS Cloud Providers through the Barter of Resources. In *Proc. of ICDCS*.

[12] Yuping Fan, Zhiling Lan, Paul Rich, William Allcock, and Michael E. Papka. 2022. Hybrid Workload Scheduling on HPC Systems. In *Proc. of IPDPS*.

[13] Yixin Bao, Yanghua Peng, and Chuan Wu. 2023. Deep Learning-Based Job Placement in Distributed Machine Learning Clusters With Heterogeneous Workloads. *IEEE/ACM Transactions on Networking* 31, 2 (2023).

[14] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and Open Problems in Federated Learning. *ACM Found. Trends Mach. Learn.* 14, 1–2 (2021).

[15] Hao Jin, Yang Peng, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2022. Federated Reinforcement Learning with Environment Heterogeneity. In *Proc. of AISTATS*.

[16] Zhijie Xie and Shenghui Song. 2023. FedKL: Tackling data heterogeneity in federated reinforcement learning by penalizing KL divergence. *IEEE JSAC* 41, 4 (2023).

[17] Jiin Woo, Gauri Joshi, and Yuejie Chi. 2023. The blessing of heterogeneity in federated Q-learning: linear speedup and beyond. In *Proc. of ICML*.

[18] Chenyu Zhang, Han Wang, Aritra Mitra, and James Anderson. 2024. Finite-Time Analysis of On-Policy Heterogeneous Federated Reinforcement Learning. *arXiv* (2024).

[19] Han Wang, Sihong He, Zhili Zhang, Fei Miao, and James Anderson. 2024. Momentum for the Win: Collaborative Federated Reinforcement Learning across Heterogeneous Environments. *arXiv* (2024).

[20] Zheyi Chen, Jia Hu, Geyong Min, Chunbo Luo, and Tarek El-Ghazawi. 2022. Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning. *IEEE TPDS* 33, 8 (2022).

[21] Tiangang Li, Shi Ying, Yishi Zhao, and Jianga Shang. 2024. Batch Jobs Load Balancing Scheduling in Cloud Computing Using Distributional Reinforcement Learning. *IEEE TPDS* 35, 1 (2024).

[22] Rui Han, Shilin Wen, Chi Harold Liu, Ye Yuan, Guoren Wang, and Lydia Y. Chen. 2022. EdgeTuner: Fast Scheduling Algorithm Tuning for Dynamic Edge-Cloud Workloads and Resources. In *Proc. of INFOCOM*.

[23] Yunfeng Zhao, Chao Qiu, Xiaoyun Shi, Xiaofei Wang, Dusit Niyato, and Victor C. M. Leung. 2024. Cur-CoEdge: Curiosity-Driven Collaborative Request Scheduling in Edge-Cloud Systems. In *Proc. of INFOCOM*.

[24] Alysa Ziying Tan, Han Yu, Lizhen Cui, and Qiang Yang. 2023. Towards Personalized Federated Learning. *IEEE TNNLS* 34, 12 (2023).

[25] Leming Shen and Yuanqing Zheng. 2023. FedDM: Data and Model Heterogeneity-Aware Federated Learning via Dynamic Weight Sharing. In *Proc. of ICDCS*.

[26] Jiechao Gao, Wenpeng Wang, Fateme Nikseresht, Viswajith Govinda Rajan, and Bradford Campbell. 2023. PFDRL: Personalized Federated Deep Reinforcement Learning for Residential Energy Management. In *Proc. of ICPP*.

[27] Tai Manh Ho, Kim-Khoa Nguyen, and Mohamed Cheriet. 2022. Federated Deep Reinforcement Learning for Task Scheduling in Heterogeneous Autonomous Robotic System. In *Proc. of GLOBECOM*.

[28] 2014. Google 2011 Workloads Archive. https://github.com/google/cluster-data/. Accessed in July, 2024.

[29] 2019. Alibaba Workloads Archive. https://github.com/alibaba/clusterdata/. Accessed in July, 2024.

[30] 2019. Chameleon Cloud Traces. https://www.sciencecloud.org/cloud-traces/. Accessed in July, 2024.

[31] Viktória Spišaková, Dalibor Klusáček, and Lukáš Hejtmánek. 2023. Using Kubernetes in Academic Environment: Problems and Approaches. In *Proc. of JSSPP*.

[32] Dalibor Klusáček and Boris Parák. 2018. Analysis of Mixed Workloads from Shared Cloud Infrastructure. In *Proc. of JSSPP*.

[33] 2021. JSSPP 2021 Workloads Archive. https://jsspp.org/workload/. Accessed in July, 2024.

[34] Meirui Jiang, Anjie Le, Xiaoxiao Li, and Qi Dou. 2024. Heterogeneous Personalized Federated Learning by Local-Global Updates Mixing via Convergence Rate. In *Proc. of ICLR*.

[35] Yishay Mansour, Mehryar Mohri, Jae Ro, and Ananda Theertha Suresh. 2020. Three Approaches for Personalization with Applications to Federated Learning. *arXiv* (2020).

[36] Truc Nguyen and My T. Thai. 2024. Preserving Privacy and Security in Federated Learning. *IEEE/ACM Transactions on Networking* 32, 1 (2024).

[37] Junqing Le, Di Zhang, Xinyu Lei, Long Jiao, Kai Zeng, and Xiaofeng Liao. 2023. Privacy-Preserving Federated Learning With Malicious Clients and Honest-but-Curious Servers. *IEEE TIFS* 18 (2023).

[38] Yifeng Zheng, Shangqi Lai, Yi Liu, Xingliang Yuan, Xun Yi, and Cong Wang. 2022. Aggregation service for federated learning: An efficient, secure, and more resilient realization. *IEEE TDSC* 20, 2 (2022).

[39] Simon Queyrut, Valerio Schiavoni, and Pascal Felber. 2023. Mitigating adversarial attacks in federated learning with trusted execution environments. In *IEEE ICDCS*.

[40] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. 2021. Federated Reinforcement Learning: Techniques, Applications, and Open Challenges. *arXiv* (2021).

[41] Muhammed Tawfiqul Islam, Shanika Karunasekera, and Rajkumar Buyya. 2022. Performance and Cost-Efficient Spark Job Scheduling Based on Deep Reinforcement Learning in Cloud Computing Environments. *IEEE TPDS* 33, 7 (2022).

[42] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource Management with Deep Reinforcement Learning. In *Proc. of HotNets*.

[43] Othmane Marfoq, Giovanni Neglia, Richard Vidal, and Laetitia Kameni. 2022. Personalized Federated Learning through Local Memorization. In *Proc. of ICML*.

[44] Zichen Ma, Yu Lu, Wenye Li, Jinfeng Yi, and Shuguang Cui. 2021. PFedAtt: Attention-based Personalized Federated Learning on Heterogeneous Clients. In *Proc. of ACML*.

[45] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv* (2016).

[46] Sheng Yue, Xingyuan Hua, Lili Chen, and Ju Ren. 2024. Momentum-Based Federated Reinforcement Learning with Interaction and Communication Efficiency. In *Proc. of INFOCOM*.

[47] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv* (2017).