RESEARCH ARTICLE

Job Scheduling in Hybrid Clouds With Privacy Constraints: A Deep Reinforcement Learning Approach

Haoyang He[1] | Yan Gu[1] | Qingzhi Liu[2] | Hao Wu[3] | Long Cheng[1]

[1]State Key Laboratory of Alternate Electrical Power System With Renewable Energy Sources, School of Control and Computer Engineering, North China Electric Power University, Beijing, China | [2]Information Technology Group, Wageningen University and Research, Wageningen, The Netherlands | [3]Computer Science Department, Maynooth University, Maynooth, Ireland

Correspondence: Long Cheng (lcheng@ncepu.edu.cn)

Received: 8 February 2024 | **Revised:** 19 September 2024 | **Accepted:** 24 September 2024

Funding: This work was supported by Fundamental Research Funds for the Central Universities (2023YQ002).

Keywords: cloud computing | deep reinforcement learning | hybrid cloud | job scheduling | privacy preserving

ABSTRACT

With the proliferation of cloud computing and the escalating demand for extensive data processing capabilities, an increasing number of enterprises are embracing hybrid cloud solutions. However, as more businesses move toward hybrid clouds, the need for effective solutions to privacy and security concerns becomes increasingly important. Although current scheduling approaches for cloud computing have addressed privacy protection to some extent, few have adequately considered the unique challenges posed by hybrid clouds. To address this gap, we propose a novel approach for scheduling jobs in hybrid clouds that prioritizes privacy protection. Our approach, called PH-DRL, leverages Deep Reinforcement Learning (DRL) to intelligently allocate jobs to virtual machines, optimizing both privacy and Quality of Service (QoS), while minimizing response time. We present the detailed implementation of our approach and our experimental results demonstrate the superior performance of PH-DRL in terms of privacy protection compared to existing methods.

1 | Introduction

The cloud computing model, often referred to as the service-based model, is witnessing a substantial surge in adoption due to its ability to provide instant access to a shared pool of computing resources [1]. Popular cloud service providers like Amazon Web Services (AWS) and Microsoft Azure provide instances that can be utilized on a pay-as-you-go basis, giving users the flexibility to choose the resources that meet their specific requirements [2]. This model's availability, scalability, and ease of use have established it as a preferred solution for addressing the computing requirements of large volumes of data generated by end users, and many works have demonstrated the benefits of cloud computing for data processing [3].

In general, cloud computing environments can be broadly classified into three categories based on their deployment methods: public, private, and hybrid cloud [4]. The private clouds provide a high degree of security and privacy while public clouds typically provide low-cost access to computing resources, with various pay-per-use models [5]. The hybrid cloud architecture traditionally involves the integration of on-premises, private, and public cloud services through orchestration across the different cloud platforms. It combines the controllability of private clouds and the resources scalability of public clouds, providing elastic resource allocation, distributed deployment, and proximity-based access. Therefore, the hybrid cloud is a more flexible and cost-effective solution. The general architecture of a hybrid cloud computing environment is shown in Figure 1.

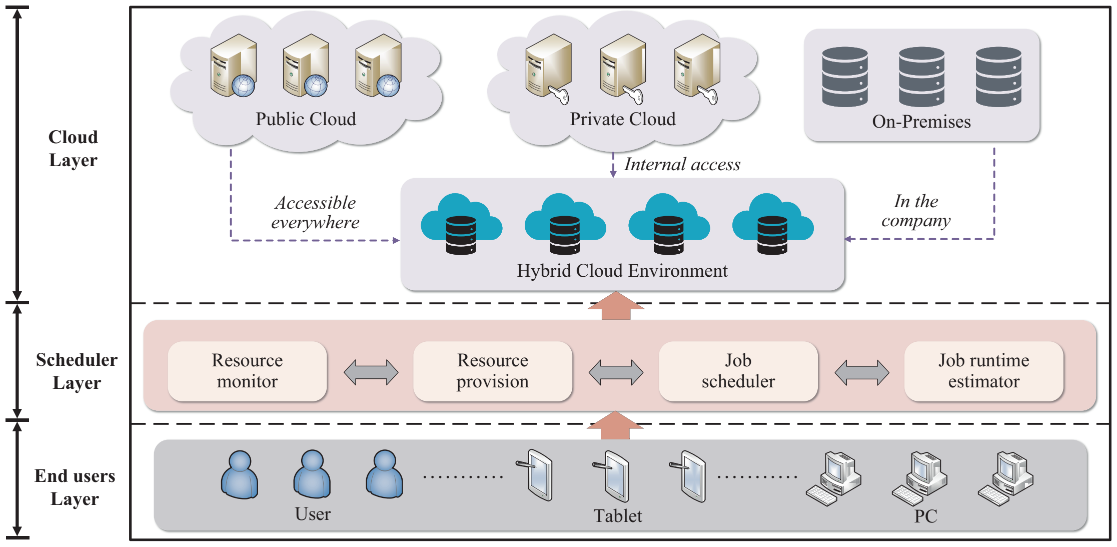

FIGURE 1 | The general architecture of a hybrid cloud computing environment.

This figure illustrates how different types of infrastructure are combined into a single, heterogeneous environment. Private clouds provide advanced security controls, but may result in lower resource utilization. Conversely, public clouds offer greater elasticity and computing power, but introduce security risks, especially for sensitive data and privacy. Therefore, it is crucial to srike the trade-off between privacy and security objectives as well as computational efficiency to achieve optimal performance.

In the realm of hybrid cloud computing, job scheduling is a well-researched area. Applications in cloud environments typically receive a constant stream of jobs from users, making it essential to have an intelligent scheduler that can effectively allocate these jobs to the appropriate resources [6]. Job scheduling enables the optimization of resource utilization, facilitates elastic scalability, load balancing, and ensures data security. During such a process, the communication of job scheduling is vulnerable to attacks due to factors such as the fragility of communication links, data risks, and the security of third-party services and integrations. Especially, privacy concerns have emerged as a major issue. For example, location-based services (LBS) provided by mobile apps, are particularly vulnerable to privacy breaches. Although LBS can provide users with convenience, they also pose a significant threat to their privacy. In the case that computing jobs with location information are scheduled to public clouds, malicious actors may be able to deduce sensitive information, such as users' habits, preferences, and even real identities, which can lead to privacy violations and other security risks. To remedy this problem, one potential solution is to limit the processing of privacy data to specific, trusted clouds, [7] that is to say, the computing jobs from users are dispatched to a hybrid cloud environment. Although some studies have proposed hybrid cloud job scheduling strategies, they tend to focus on the optimization of cost or response time, and few studies have considered privacy in their approaches [8].

In current advanced job scheduling frameworks in the hybrid cloud environment, the majority simply assign privacy-sensitive tasks to trusted virtual machines (VMs), without performing further categorization of these tasks [7, 9]. In this paper, we propose a three-layer privacy model that classifies jobs into three types according to privacy levels. To balance privacy protection with other user requirements during job execution in a hybrid cloud, we propose a novel scheduling method called PH-DRL, which incorporates the proposed privacy model, on the basis of deep reinforcement learning [10] (DRL). Specifically, by using DRL, we are able to effectively handle real-time job scheduling in a hybrid cloud environment in an intelligent way, resulting in improved performance and superior decision-making capabilities. The novelty of this paper is to combine a three-layer privacy model with DRL for intelligent decision-making in job scheduling for hybrid cloud environments. The main objective is to minimize response time while ensuring privacy and meeting quality of service (QoS) constraints. We present the detailed implementation of our method, and our experimental results demonstrate the effectiveness as well as superiority of our proposed PH-DRL method.

In general, the main contributions of this paper are as follows:

- To address the challenges of job scheduling in a hybrid cloud environment while considering privacy constraints, we propose a new approach called PH-DRL. This approach is based on DRL which enables the scheduler to learn how to select VMs that meet users' privacy needs while also ensuring high QoS.
- We provide a detailed description of our DRL model for job scheduling in hybrid clouds. The model takes into account the privacy constraints, QoS of job execution and job response time.
- We evaluate the performance of PH-DRL using experiments and compare it with existing job scheduling approaches. The results show that PH-DRL outperforms other methods in terms of job assignment success rate and response time.

The rest of this paper is organized as follows: We review the relevant literature in Section 2. We provide an overview of our proposed system architecture in Section 3. We present the details of our DRL approach in Section 4 and evaluate the performance of our approach through experiments in Section 5. Finally, we conclude our work in Section 6.

2 | Related Works

Job scheduling in clouds stands as a long-standing issue within the realm of cloud computing optimization, with various solutions proposed over the past decades. For instance, the improved multi-verse optimizer (EMVO) algorithm [11] has been developed to effectively reduce makespan and increase resource utilization in the cloud computing environment. Another solution, the modified Henry gas solubility optimization (HGSO) algorithm, [12] based on the whale optimization algorithm (WOA) and comprehensive opposition-based learning (COBL), aims to optimize job scheduling in cloud computing. Additionally, a fuzzy-based job scheduling method (SAEA) [13] has been devised to address multi-objective optimization scheduling problems, considering factors such as energy costs, degree of load imbalance, and security level. However, most of these methods are designed to handle batch jobs and are not well-suited for real-time workloads. In contrast, this paper focuses on addressing the challenge of real-time job scheduling in the cloud environment.

In recent years, the hybrid cloud architecture has become increasingly popular, due to its ability to meet performance and cost requirements. Specifically, there have been some studies on job scheduling algorithms for the hybrid cloud [14]. These studies have mainly focused on optimizing execution cost and job response times in hybrid cloud environments. For example, Yuan, Bi, and Zhou [15] proposed a temporal job scheduling algorithm for scheduling jobs within their delay constraints, taking into account temporal variations in the green hybrid cloud. Additionally, Zhu et al. [16] introduced a scheduling approach called matching and multi-round allocation (MMA), which aims to maximize makespan and overall cost for all submitted jobs while taking security and dependability considerations into account in a multi-cloud environment. Our research focuses on hybrid cloud job scheduling. In the meantime, we aim to develop a scheduling model based on DRL with neural networks to solve optimization problems in hybrid clouds.

In relation to DRL, the approach has been increasingly used to tackle complex decision-making problems in various research fields, due to its powerful ability to make decisions without requiring prior knowledge [10, 17]. For instance, Wurman et al. [18] used DRL to train agents to beat a human expert in the game Gran Turismo. Liu et al. [19] presented a DRL-based Internet of Things (IoT) network dynamic clustering solution. Additionally, James, Yu, and Gu [20] and Zhu et al. [21] proposed different combinatorial optimization strategies using DRL to solve the vehicle trip generation problem and the UAV trajectory generation problem, respectively. In fact, DRL has also been applied to cloud job scheduling problems due to its powerful decision-making capabilities. For example, the DRL-based algorithm proposed by Wei et al. [6] can guarantee the QoS requirements of users while making cloud job scheduling under different real-time changing workloads. Huang et al. [22] provided an advanced DRL-based method for cloud job scheduling called AIRL, which can outperform existing scheduling approaches in terms of QoS and average response time. Gu et al. [23] utilized the DRL and simulated annealing to determine the optimal scheduling strategy, aiming to balance the response time and cost. Furthermore, Cheng et al. [24] considered not only the QoS requirements of users but also the cost associated with executing jobs on the virtual instances through DRL. In comparison, our research focuses on improving privacy protection while maintaining high QoS for real-time jobs in a hybrid cloud environment.

Generally, privacy has been a concern in the area of cloud job scheduling, and many studies have tried to address the issue in an effective way [25]. Moreover, the privacy issue has also been considered in hybrid cloud job scheduling, with some research focusing on optimizing privacy in workflow scheduling. In fact, the problem is quite challenging as workflows have dependencies between jobs. For example, Wen et al. [9] proposed the MOPA algorithm, which stands for Multi-Objective Privacy-Aware Workflow Scheduling, to solve critical privacy disclosure issues in Cyber-Physical-Social environments. Additionally, Lei, Wu, and Xu [26] investigated ways to significantly reduce the monetary cost when developing workflows in the hybrid cloud with deadlines and privacy restrictions. Furthermore, Sharif et al. [27] proposed a new SaaS scheduling broker made of MPHC-P1, MPHCP2, and MPHC-P3 policies to protect privacy while scheduling workflow jobs in hybrid clouds to meet customer deadlines. Our work focuses on developing a hybrid cloud job scheduling model based on DRL that takes privacy considerations into account while ensuring high QoS for real-time jobs.

Recent research has increasingly focused on optimizing job scheduling in hybrid cloud environments, targeting objectives such as minimizing financial costs [28] and reducing makespan [29]. However, few studies have considered privacy constraints in the optimization problem of job scheduling in hybrid clouds. In this paper, we propose PH-DRL, a novel approach to solve privacy protection issues in the hybrid cloud environment. Our approach aims to balance the trade-off between privacy protection and other objectives such as cost, performance, and makespan.

Based on above discussion, as shown in Table 1, we summarize the main features of some typical job scheduling works. Specifically, some works are in a single cloud environment, while PH-DRL can conduct job scheduling in a hybrid cloud environment. Then, the majority of existing works are concerned with scheduling jobs in batch form, while PH-DRL is capable of scheduling real-time jobs. Additionally, in contrast to PH-DRL, which considers both response time and privacy constraints as optimization objectives, most extant works tend to regard only one aspect, or even none.

3 | System Architecture of PH-DRL

3.1 | Hybrid Cloud Computing Model

As mentioned above, Figure 1 illustrates the general architecture for job scheduling in hybrid cloud. The job processing can

TABLE 1 | Main features of typical approaches in literature.

Ref.	Scheduling type real-time	Environment hybrid cloud	Constraint privacy	Optimization model DRL	Optimization objectives
Wei et al. [6]	✓	—	—	✓	Minimizing response time
Yuan, Bi, and Zhou [15]	—	✓	—	—	Maximizing profit
Zhu et al. [16]	—	✓	—	—	Minimizing makespan and total cost
Baniata et al. [25]	—	—	✓	—	Minimizing task computing time
Wen et al. [9]	—	—	✓	—	Minimizing execution time and monetary cost
Lei et al. [26]	—	✓	✓	—	Minimizing monetary cost
Chen et al. [28]	—	✓	—	—	Minimizing makespan and monetary cost
Zhang et al. [29]	—	✓	—	—	Minimizing makespan
PH-DRL	✓	✓	✓	✓	Minimizing response time

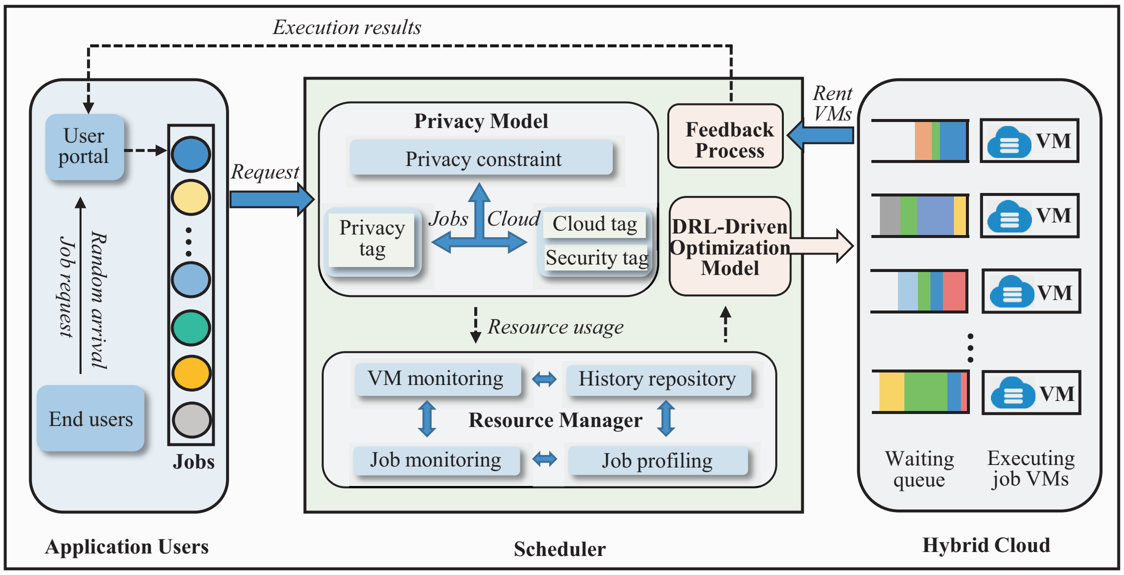

FIGURE 2 | The privacy-aware hybrid cloud scheduling framework using DRL.

be divided into three phases: transmission, scheduling, and execution. While the performance model of data transmission in job scheduling has been thoroughly studied in previous works, [15, 30] our focus in this work is to design a new scheduling approach that prioritizes privacy. Therefore, we will only consider the scheduling and execution phases in the following. Specifically, in our PH-DRL, we utilize virtual instances from various IaaS providers to host applications. This allows for flexibility in terms of utilizing both private and public cloud resources. Moreover, for a general case, users can submit requests with defined QoS requirements, such as response time, computational power, and resource efficiency.

Generally, the proposed PH-DRL framework, as illustrated in Figure 2, includes three key components: the user portal, job scheduler, and hybrid cloud environment. The user portal serves as a liaison between the scheduler and end-users, facilitating the submission of job requests and the dissemination of job execution results. The job scheduler, which includes the DRL model, privacy model, and resource manager, is responsible for making job assignment decisions and selecting the appropriate virtual machine for execution. The DRL model and privacy model work together to perform job filtering and assignment based on privacy rules. Moreover, the resource manager oversees the performance of other components and utilizes various functions to improve job scheduling. These functions include VM monitoring, job monitoring, history repository, and job profiling. VM monitoring constantly supervises the performance of VMs, while job monitoring tracks the assignment and execution of jobs in the hybrid cloud environment. The history repository stores historical files of job execution and virtual machine performance, and job profiling is used to identify characteristics of jobs such as job types and execution length.

To facilitate the description of the optimization problem studied in this paper, we provide the mathematical definitions of the privacy model, workloads, cloud resources, and the job scheduling mechanism as follows. The notations we used are given in Table 2.

TABLE 2 | Notations used in our system.

Notation	Meaning
jID_i	The id of a job
$arrivalT_i$	The time point of job arrival
$jType_i$	The type of a job
$jSize_i$	The length of a job
QoS_i	The QoS requirements of a job
PT_i	The privacy tag of a job
Vid_j	The id of VM
Vpp_j	The cloud tag of VM
$Vtype_j$	The type of VM
V^p	Processing speed (instructions per second)
V^p_{com}	Processing speed for computing intensive jobs
V^p_{io}	Processing speed for I/O intensive jobs
$Vsecurity_j$	The security tag of VM
T_i	Job's response time
T_i^{exe}	Job's execution time
T_i^{wait}	Job's waiting time
V_j^{cur}	The available time point of VM
T_i^{leave}	The completion time of job

3.2 | Privacy Model in Hybrid Cloud Environment

When executing a job from end users in a hybrid cloud environment, privacy-sensitive data and jobs are unable to be exposed to public clouds, [26] and we can assign these jobs to private clouds, as private clouds typically outperform public clouds in controllability and privacy protection [27]. In order to distinguish whether a VM is from a private cloud or a public cloud, we use a cloud tag described as $Vpp = \{VM_1, VM_2\}$, where VM_1 and VM_2 represent the VMs from private clouds and public clouds, respectively.

Compared with the cloud environments that most researchers have focused on, the hybrid cloud environment in our study considers an additional aspect, that is privacy constraint. Exposing privacy-sensitive data to public clouds introduces several risks, including unauthorized access, data breaches, and compliance issues with regulations. Public clouds, while offering scalability and cost benefits, often lack the stringent security measures required for handling sensitive data. Therefore, to mitigate these risks, our approach ensures that privacy-sensitive data is processed within private cloud environments. It is worth mentioning that, when designing the privacy levels of our job scheduling rules, data security is regarded as a supporting condition. The security and bandwidth of data transmission should be guaranteed when transferring across cloud platforms over the Internet against data breaches or data theft. Therefore, we also consider security in the privacy model, which includes confidentiality and integrity.

Based on this, a three-level privacy model is designed to ensure privacy and security in the job scheduling process. Specifically, jobs in the hybrid cloud environment can be divided into three categories according to privacy levels, and for ease of representation, we define privacy tag $PT = \{L_1, L_2, L_3\}$, where (1) L_1 denotes jobs that can only be assigned to the private cloud; (2) L_2 represents jobs that can be assigned to either private clouds or public clouds which has guaranteed security; and (3) L_3 represents jobs that can be assigned to the private cloud and the public cloud without any restriction. Moreover, in contrast to VMs in a typical cloud environment, VMs in the hybrid cloud environment increase security tags as $Security = \{t_1, t_2\}$, where t_1 represents that the VM satisfies the security requirements mentioned above, and t_2 denotes that the VM does not meet the security requirements. The design of our privacy model is similar to that in recent works [26, 27]. It can be extended based on specific needs by integrating additional techniques, such as updating security tags during scheduling, handling data breaches, reacting to changes in the security posture of VMs, or incorporating thread injection when external vulnerabilities are present within the hybrid cloud. Nonetheless, it should be noted that these techniques are supplementary to our approach, and a detailed integration and analysis of them falls outside the scope of our study.

We record the scheduling rules of our privacy model in Table 3. In each cell, symbol "✓" signifies a rule-compliant assignment, and symbol "×" denotes a non-conforming assignment.

As shown in Table 3, $VM_1(t_1)$ indicates the VM from the private cloud with security; and $VM_1(t_2)$ represents the VM from the private cloud without security. Similarly, $VM_2(t_1)$ denotes the VM from the public cloud with security; and $VM_2(t_2)$ represents the VM from the public cloud without security. For example, we consider a scenario where the private cloud consists two VMs, VM_1 and VM_2, while the public cloud has two VMs, VM_3 and VM_4. We assume that VM_1 and VM_3 provide guaranteed security. Consequently, during the job scheduling process, jobs with L_1 tag can be scheduled to VM_1 and VM_2, jobs with L_2 tag can be scheduled to VM_1 and VM_3, and jobs with L_3 tag can be scheduled to any VM.

3.3 | Workloads Characteristics

In our scheduling model, for the sake of simulating the real-time computing jobs in hybrid cloud environments, we set them to arrive randomly and with an uncertain pattern. Additionally, we consider jobs to be independent and non-interfering with each other. When submitting these jobs, users can specify their QoS needs, which can be regarded as the criterion to judge the success of the scheduling. We consider that job scheduler can make decisions as soon as the job arrives. Jobs can be categorized into various types, including computing intensive, I/O intensive, memory intensive, and hybrid type. Among them, the first two are the most typical types. Specifically, computing intensive jobs require

TABLE 3 | Scheduling rules of privacy model.

	L_1	L_2	L_3
$VM_1(t_1)$	✓	✓	✓
$VM_1(t_2)$	✓	×	✓
$VM_2(t_1)$	×	✓	✓
$VM_2(t_2)$	×	×	✓

a significant amount of processing power and computing time to complete. On the other hand, I/O intensive jobs involve a substantial number of input/output operations, such as reading from or writing to disk. Without loss of generality, in this paper, jobs are divided into two typical types according to the dependence on different types of the resource, namely computing intensive jobs and I/O intensive jobs. Thus, each job can be described as:

$$Job_i = \left\{jID_i, T_i^{at}, jType_i, jSize_i, QoS_i, PT_i\right\} \tag{1}$$

where jID_i is the id of the job, which represents the exclusive identifier of the job. T_i^{at} denotes the time point of job arrival. $jType_i$ is the type of the job. $jSize_i$ is the length of the job, which means the required number of instructions to handle the job. QoS_i is the QoS requirements of the job, which is specifically the response time requirements of the user for the submitted jobs. PT_i is the privacy tag of the job, which is used to indicate the privacy level of the job.

3.4 | Cloud Resource Model

The job scheduling model proposed in this paper is in a hybrid cloud environment, which leverages the computing power of both private clouds and public clouds. Similar to the job workloads, it is general for VMs in the cloud computing market to have two types, that is, computing intensive type and I/O intensive type, which indicates that a job could have various execution times on different types of VMs. Moreover, since we focus on privacy protection in the hybrid cloud, it is necessary to take into account the cloud tag and the privacy tag mentioned above for VMs. Thus, a VM in the hybrid cloud environment can be described as:

$$V_j = \left\{Vid_j, Vpp_j, Vtype_j, V^p_{\mathrm{com}_j}, V^p_{\mathrm{io}_j}, Vsecurity_j\right\} \tag{2}$$

where Vid_j is the VM id, indicating the special identifier of the VM. Vpp_j is the cloud tag to identify whether a VM is from a private cloud or a public cloud. $Vtype_j$ is the type of the VM. $V^p_{\mathrm{com}_j}$ and $V^p_{\mathrm{io}_j}$ are the processing capabilities of the VM for executing computing intensive jobs and I/O intensive jobs, respectively. $Vsecurity_j$ is the security tag of the VM, which is used to determine whether a VM meets security requirements.

3.5 | Mechanisms for Scheduling Jobs

The job scheduler in PH-DRL framework is responsible for assigning jobs to specific VMs. There exists a waiting queue for each VM, and if the VM is unavailable when a job is allocated to it, the job will be placed on the waiting queue of the VM. In our proposed model, the jobs follow a first-come-first-serve (FCFS) manner, in addition, the execution of them are non-preemptive, which means that each VM can merely execute one job at a time. The job response time is composed of two parts: job execution time and waiting time, [6] which can be described as:

$$T_i = T_i^{\mathrm{exe}} + T_i^{\mathrm{wait}} \tag{3}$$

where T_i is the response time of the job, T_i^{exe} is the execution time of the job, and T_i^{wait} is the waiting time of the job. During the execution time, the operation involves completing the job. We can further describe the execution time T_i^{exe} as:

$$T_i^{\mathrm{exe}} = \begin{cases} \frac{jSize_i}{V^p_{\mathrm{com}_j}}, & \text{if } job_i \text{ is computing intensive} \\ \frac{jSize_i}{V^p_{\mathrm{io}_j}}, & \text{if } job_i \text{ is I/O intensive} \end{cases} \tag{4}$$

where $jSize_i$ is the length of the job.

Moreover, when a job is assigned to a VM, the VM may be idle or busy at the current time. We make the rule that if the VM is idle, the current job can be executed immediately, while if the VM is busy, the current job needs to wait and cannot be executed until the jobs in the waiting queue have finished. We define V_j^{cur} to represent the available time point, which means that the time point when the V_j can perform the current job. Thus, we describe the waiting time T_i^{wait} as:

$$T_i^{\mathrm{wait}} = \begin{cases} 0, & \text{if queue } L_j^i = 0 \\ V_j^{\mathrm{cur}} - T_i^{at}, & \text{else} \end{cases} \tag{5}$$

where queue L_j^i is the waiting queue of V_j when the current job arrives, V_j^{cur} is the available time point of V_j, and T_i^{at} is the time point of the current job arrival.

Further, we define Job_i to signify the newly arriving job, and $Job_{i'}$ to denote the job assigned to V_j before Job_i. Thus, the available time point of V_j can be described as:

$$V_j^{cur} = T_{i'}^{\mathrm{wait}} + T_{i'}^{at} + T_{i'}^{\mathrm{exe}} \tag{6}$$

where V_j^{cur} is the available time point of V_j, $T_{i'}^{\mathrm{wait}}$ is the waiting time of the job $Job_{i'}$, $T_{i'}^{at}$ is the arrival time of job $Job_{i'}$, and $T_{i'}^{\mathrm{exe}}$ is the execution time of the job $Job_{i'}$.

We consider that the completion time point of job Job_i is subject to its arrival time point and response time. Thus, the completion time of job Job_i can be represented as:

$$T_i^{leave} = T_i^{at} + T_i \tag{7}$$

where T_i^{leave} denotes the completion time of job Job_i, T_i^{at} is the time point of job Job_i arrival, and T_i is the response time of the job Job_i.

3.6 | Success Condition for Scheduling

We consider that there are two main criteria for judging the success of scheduling. First, as mentioned above, we focus on privacy issues in PH-DRL framework, so the process of the job scheduling needs to satisfy privacy constraints. Second, in order to maintain the QoS requirements provided by the users, we have to guarantee that the job is executed within the expected time. Therefore, we define a success criterion for our scheduling method as:

$$success(job_i, VM_j) = \begin{cases} 1, & \text{if } T_i \leqslant QoS_i \text{ and } P = 1 \\ 0, & \text{else} \end{cases} \tag{8}$$

where T_i is the response time of the job, and QoS_i is the QoS requirement of the job provided by the user. P is used to denote whether the scheduling rules of the privacy model are satisfied. If the rules are followed, $P = 1$; otherwise, $P = 0$.

Based on these descriptions and definitions, the job-scheduling problem in our work can be succinctly described as: in the hybrid cloud environment, the objective is to allocate the jobs to a set of VMs with minimum response time T_i and maximize success rate *success* of the scheduling model considering privacy constraints, which can be defined as:

$$
\begin{aligned}
&Minimize T_i = T_i^{\text{exe}} + T_i^{\text{wait}} \\
&Maximize\, success(job_i, VM_j) = \begin{cases} 1, & \text{if } T_i \leqslant QoS_i \text{ and } P = 1 \\ 0, & \text{else} \end{cases}
\end{aligned}
\tag{9}
$$

4 | The Implementation of PH-DRL

We focus on utilizing DRL as a system controller to ensure privacy-aware job scheduling in a hybrid cloud environment. Specifically, we have adopted the deep Q-network (DQN) in our implementation. In this section, we introduce DRL basics and the details of DRL-based job scheduling.

4.1 | Deep Reinforcement Learning Basics

DRL is a powerful approach for solving complex control and optimization problems. DRL consists of five fundamental components: the action space, state space, reward function, agent, and environment. In DRL, Markov Decision Process (MDP) serves as a formal description and modeling tool for the environment [31]. In this paper, we construct the job scheduling problem as a MDP and utilize the interaction between the agent and the environment to continuously learn and adjust strategies based on real-time feedback, aiming to achieve better resource utilization and performance optimization. For a general case, the interaction process proceeds as follows: at each time slot t, the agent observes the environment and obtains a state s_t. It then executes an action a_t based on the observed state, and receives a reward r_t according to the selected action and the reward function. After receiving the reward, the agent perceives the next state s_{t+1}, and the process repeats until the final state is reached. The agent aims to find an optimal policy that maximizes the reward by considering all states. To achieve this, DRL establishes an association between each state-action pair using a deep neural network (DNN) [32]. The DQN is commonly used as the foundation for the deep Q-learning framework, which helps the agent make decisions about possible actions. Figure 3 shows the idea of the PH-DRL framework. Specifically, jobs are requested by end users. Subsequently, the attributes of the current job and the information of the VMs are converted into state vectors. The agent interacts with the environment to make intelligent decisions, assigning each job to the most suitable VM. Finally, the execution results are returned to the application users.

4.2 | DRL-Based Job Scheduling

In the hybrid cloud environment, the complexity of users' jobs can result in a large state space for the DRL model, leading to a significant training time. To overcome this challenge, we have incorporated an event-driven decision-making mechanism. This mechanism allows for immediate scheduling decisions when new jobs arrive. Additionally, by following the FCFS job arrival manner, we can reduce the number of possible actions. The action represents the target VM instance for the current job at each decision point. The proposed DRL-based job scheduling

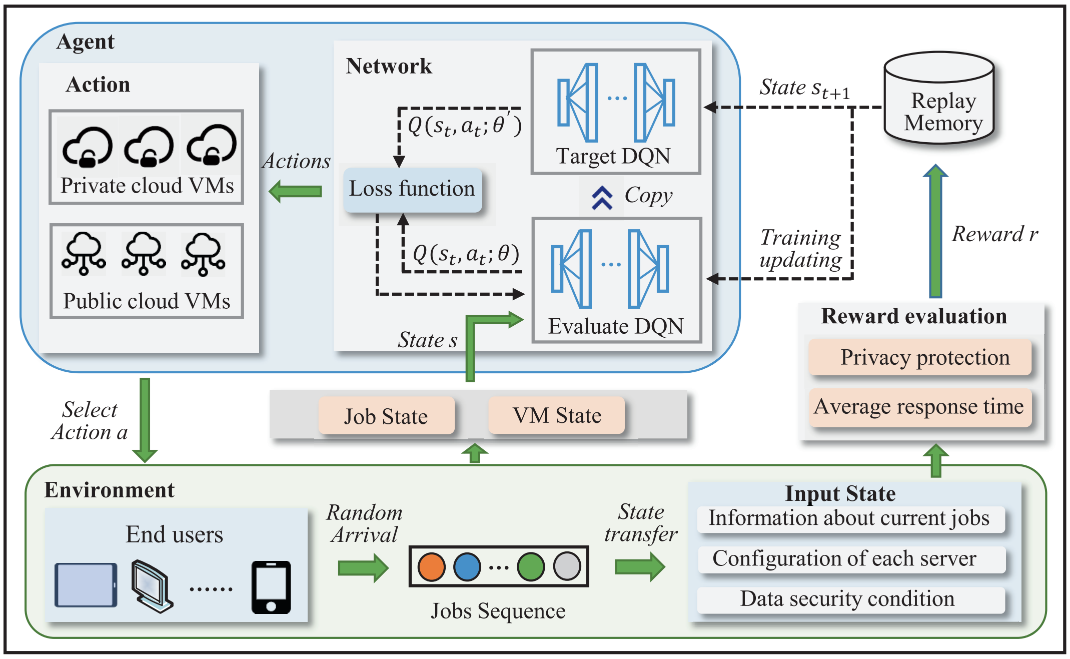

FIGURE 3 | The operating procedure of the PH-DRL framework.

process comprises two distinct phases: the decision-making phase and the model training phase.

4.2.1 | Decision-Making Phase

The decision-making phase is responsible for scheduling the current job to the suitable VM instances in the hybrid cloud environment through deep Q-learning techniques. By observing the current state of the environment, the scheduler estimates q-values of all actions using DNN at each decision point. According to the estimated q-values, a VM instance will be selected to execute the job, and the agent will get a reward corresponding to this scheduling decision. It is capable of adapting to dynamic changes in job characteristics and environmental information, resulting in optimal real-time decision-making. The important components of our DRL model is described as below.

Action space ($\mathscr{A}$): The set of all possible actions available to an agent in a specific environment is known as an action space. Considering the hybrid cloud environment, the action space consists of the leased VM instances from both private clouds and public clouds. Hence, the action space in our DRL model is defined as:

$$\mathcal{A} = [VM_1 \cup VM_2] \tag{10}$$

where VM_1 and VM_2 represent the private VMs and public VMs in the hybrid cloud environment, respectively.

State space ($\mathcal{S}$): All possible states that an agent can transition based on actions are contained in the state space. In our scenario, assuming a new job arrives at time t, the state space of our DRL model at time t can be defined as follows:

$$\mathcal{S} = \left[S_{job} \cup S_{VM}\right] \tag{11}$$

where S_{job} is the state of the current job at time t, and S_{VM} is the state of VMs at time t. More specifically, the entire state space can be described as:

$$\mathcal{S} = \left[jType_j, jSize_j, QoS_j, PT_j, T_{j1}^{\text{wait}}, T_{j2}^{\text{wait}}, \ldots, T_{jN}^{\text{wait}}\right] \tag{12}$$

where $jType_j$, $jSize_j$, QoS_j, and PT_j represent the type, length, quality of service, and privacy tag of the current job respectively, and T_{jN}^{wait} is the waiting time for the job in N^{th} VM.

Reward ($\mathcal{R}$): Once taking the action a_t in the current state s_t, the system will evolve into a new state s_{t+1} and receive a reward r_t from the environment. The reward plays a crucial role in guiding the agent's decision-making process. With the objective of satisfying the privacy requirements of the job in the hybrid cloud environment and ensuring the QoS requirement of the application users, we consider the effectiveness of privacy protection and average response time as the factors for influencing the reward. Thus, the reward function of our DRL model is defined as:

$$r = \left(1 + e^{1+P}\right)\left(\frac{\lambda}{T_i}\right) \tag{13}$$

where P, as mentioned above, is used to judge if the scheduling rules of the privacy model are satisfied. Specifically, if privacy requirements in scheduling process are met, then $P = 1$, which means a greater reward value; otherwise, $P = 0$, which means a smaller reward value. T_i indicates the response time of the job, and its smaller value corresponds to a larger reward value. And λ is a trade-off parameter utilized to coordinate the effect of privacy and QoS in the reward function. When the system prioritizes the impact of response time, we can increase the value of λ. Conversely, a smaller value of λ can be set when the system emphasizes privacy protection.

4.2.2 | Model Training Phase

During the model training phase, decisions and outcomes from historical job scheduling are leveraged to guide the underlying DNN in acquiring a more precise value function. Algorithm 1 illustrates the training procedure of the proposed DRL-based job scheduling model. We adopt the ϵ-greedy strategy to choose an action randomly with the possibility ϵ. As training proceeds, the value of ϵ decreases, allowing to explore actions randomly initially and gradually shift toward exploiting actions that yield higher rewards. Our training is performed offline, once the model is trained, it is used for real-time job scheduling, which has significantly lower computational requirements than training process. In the model training process, experience replay and fixed Q-targets are the two essential parts.

Experience replay [33]: At each time step, transition $\left(a_t, s_t, r_t, s_{t+1}\right)$ is stored into a replay memory Δ with capacity N_Δ. The parameter θ of DQN is updated by the minibatch which includes a fixed number of random samples S_Δ from the replay memory Δ. Two DNN neural networks have been

ALGORITHM 1 | The proposed DRL-based algorithm.

```
1: Input: initial $\epsilon$, $\alpha$, $\gamma$, learning frequency $f$, start learning time
   $\tau$, minibatch $S_\Delta$, replay period $\eta$
2: Initialize replay memory $\Delta$ with capacity $N_\Delta$
3: Initialize evaluation value function $Q$ with random parameters $\theta$
4: Initialize target value function $\widehat{Q}$ with random parameters $\theta'$
5: for each new job $j$ arrives at $t_j$ do
6:     with probability $\epsilon$, randomly choose an action; otherwise
       $A_j = \text{argmax}_A Q\left(S_j; A; \theta\right)$
7:     Schedule job $j$ according to action $A_j$, receive reward $R_j$,
       and observe state transition at next decision time $t_{j+1}$
       with a new state $S_{j+1}$.
8:     Store transition $\left(S_j, A_{j+1}, R_{j+1}, S_{j+1}\right)$ in $\Delta$
9:     if $j \geqslant \tau$ and $j \equiv 0 \bmod f$ then
10:        if $j \equiv 0 \bmod \eta$ then
11:            Reset $\widehat{Q} = Q$
12:        end if
13:        randomly select samples $S_\Delta$ from $N_\Delta$
14:        for each transition $\left(S_j, A_{j+1}, R_{j+1}, S_{j+1}\right)$ in $S_\Delta$ do
15:            $target_k = r_k + \gamma max_A Q(S_k + 1; A'; \theta')$
16:            update DNN parameters $\theta$ with loss function of
               $target_k - Q(s_k, a_k; \theta)^2$
17:        end for
18:        Gradually decrease $\epsilon$ until to the lower bound
19:    end if
20: end for
```

set up, namely, the evaluation network with parameter θ and the target network with parameter θ'. This training method outperforms the traditional Q-learning in two aspects. First, it improves data efficiency as each experience step has the potential to be repeated many times during the process of parameter update. Second, it facilitates a more efficient learning process by randomly selecting experiences for training, breaking the correlation in the training data and reducing the variance of updates.

Fixed Q-targets [34]: During the training phase of DNNs, it is common to employ both a target network and an evaluation network to mitigate parameter deviation. The two networks have identical share the same structures but have different parameters, with the target network generating target q-values and the evaluation network providing estimated q-values. To avoid divergence and oscillations during training, the parameters of the target network are periodically updated by cloning them from the evaluation network. Compared to the evaluation network's immediate parameter update, the target network's update is delayed, which helps stabilize the learning process. Overall, this approach facilitates more effective learning by reducing the impact of parameter deviation and promoting convergence toward the optimal solution.

4.2.3 | Implementation Analysis

Concerning the complexity of training in our PH-DRL, each iteration comprises two parts: the time complexity for the neural network to calculate the Q-values for VMs in the current input state, and for updating the network parameters. Given an input state size of d and the number of neurons h in the hidden layers, the time complexity of the first part is $O(d \times h^2)$. Updating the network parameters involves the use of experience replay and fixed Q-targets. For a mini-batch of size S_Δ, the time complexity for this update is $O(S_\Delta \times d \times h^2)$. Thus, the overall time complexity for each iteration of our PH-DRL is $O(S_\Delta \times d \times h^2 + d \times h^2)$, which simplifies to $O(S_\Delta \times d \times h^2)$. Assuming T iterations are required, the total time complexity of PH-DRL is $O(T \times S_\Delta \times d \times h^2)$. Upon completion of model training, the agent is capable of executing decision-making exclusively through inference. The inference time complexity is considerably lower than that of training, since PH-DRL only performs a forward propagation through the neural network to choose actions according to the current input state, with a time complexity of $O(d \times h^2)$, thus enabling the real-time assignment of arriving tasks to the appropriate VMs.

Based on the computational complexity analysis, it is evident that the training process of the PH-DRL framework is performed offline. The framework enables real-time job scheduling upon completion of model training, thereby making it suitable for large-scale hybrid cloud environments. Moreover, the PH-DRL model is designed with flexibility in mind, allowing it to be adapted to various cloud environments. By retraining the DRL model with data from different cloud configurations, the state and action spaces can be adjusted to accurately represent the specific characteristics and resource types available. This adaptability ensures that the PH-DRL approach remains effective across a range of cloud setups.

5 | Experimental Evaluation

To evaluate the performance of PH-DRL, in this section, we conducted a comparative analysis with several widely used real-time scheduling algorithms, including Random, Round-Robin, and Earliest. Additionally, we also compared PH-DRL with two state-of-the-art algorithms, namely TTS and MMA, which are hybrid cloud job scheduling approaches proposed by Yuan, Bi, and Zhou [15] and Zhu et al. [16], respectively. Our experiments were carried out using the Pytorch framework and implemented in Python 3.

5.1 | Experiment Setup

In our experiment, VMs are allocated not only to *VM*1 (private cloud) and *VM*2 (public cloud) but also categorized into two types: high-CPU and high-I/O. Similarly, job types are randomly designated as either I/O intensive and computationally intensive. It should be noted that although our proposed method considers only high-CPU intensive and high-I/O intensive job types, it has the flexibility to incorporate additional types such as memory-intensive and bandwidth-intensive depending on specific requirements. The experimental job data is generated according to specific distribution functions, where job workloads are randomly generated following a Normal distribution and job arrival times are generated according to a Poisson distribution.

The initialization of the DNN involves a neural network with 20 neurons in the hidden layers. The replay memory N_Δ is fixed at 800, and the minibatch size S_Δ is fixed at 30. The learning rate is fixed at 0.01, and the target iteration is fixed at 50 decisions per episode. Once the replay memory has enough transition samples, the DNN begins to be trained. Additionally, other parameters have been fixed as follows: $\gamma = 0.9$, $f = 1$, $\tau = 500$, and ϵ is reduced by 0.002 in each learning iteration from 0.9.

5.2 | Comparison of Scheduling Performance

5.2.1 | Varying Mean Job Arrival Rates

We compare PH-DRL with various comparison algorithms at different levels of mean job arrival rates in terms of success rate and average response time, respectively. To facilitate an intuitive comparison based on the experimental results, we plot the bar chart in Figure 4. In this experiment, the mean job arrival rates range from 10 to 30 with an interval of 5. To guarantee the fairness of the experiment, all comparison algorithms are subjected to the same experimental settings. Specifically, the percentage of high-CPU VMs and high-I/O VMs both account for 50% of the total number of VMs, similarly, both the number of secure VMs and private cloud VMs are half of all VMs. Moreover, the number of computing intensive jobs and I/O intensive jobs both account for half of the total job arrivals.

From the results in Figure 4, it is evident that PH-DRL performs significantly better than other comparison algorithms. Specifically, regardless of the number of mean job arrival rates, PH-DRL achieves an ~38% higher than the Round-Robin and Earliest method in terms of success rate. And PH-DRL also has significant

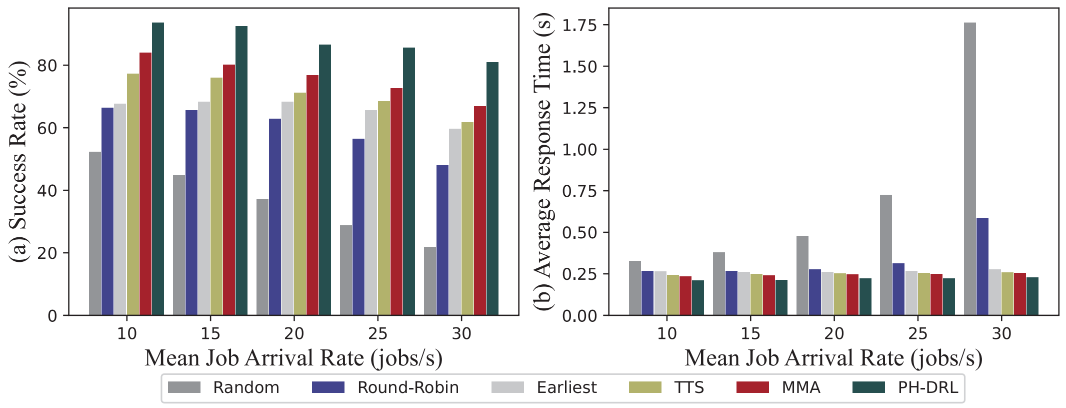

FIGURE 4 | Performance under different mean job arrival rates.

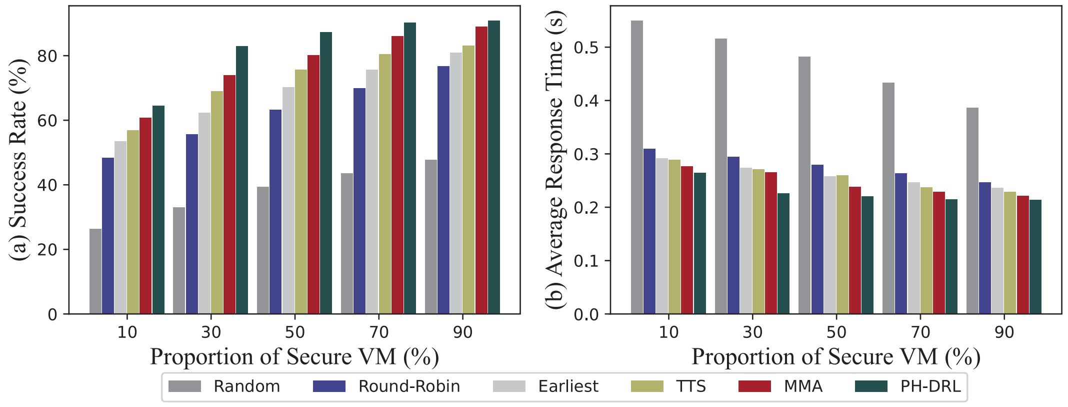

FIGURE 5 | Performance under different secure VM proportion.

advantages in success rate over the model-based methods TTS and MMA. Moreover, PH-DRL demonstrates a reduced response time in comparison to the comparison algorithms. Based on the above analysis, we conclude that PH-DRL can schedule the jobs in a hybrid cloud with a much shorter response time while having a higher success rate.

5.2.2 | Varying Secure VM Proportion

Similar to evaluation on different mean job arrival rates, in this experiment, we compare the performance of various algorithms for different proportions of secure VMs. Figure 5 shows the results of the experiment. Without loss of generality, we set the proportion of secure VMs to vary from 0.1 to 0.9 with an interval of 0.2. For all comparison algorithms, we set the proportion of computing intensive jobs and I/O intensive jobs to both 50%, the proportion of private cloud VMs and public cloud VMs to both 50%, and the mean job arrival rate to be fixed at 20.

By observing the results in Figure 5, we can conclude that PH-DRL outperforms the comparison algorithms in terms of performance. Specifically, PH-DRL achieves a success rate that is 20%–30% higher than other comparison algorithms in terms of success rate. And PH-DRL outperforms TTS and MMA by achieving better success rate. In addition, PH-DRL exhibits shorter response times compared to other comparison algorithms. Furthermore, as the proportion of secure VMs increases, we observe a continuous increase in success rate and a decrease in response time. So we draw the conclusion that PH-DRL performs better when a higher proportion of secure VMs are presented in a hybrid cloud.

5.2.3 | Varying Private Cloud VM Proportion

We conduct experiments involving different levels of proportions of private cloud VMs. The results of the experiment are displayed in Figure 6. The proportion of private cloud VMs is set to vary from 0.1 to 0.9 with an interval of 0.2. For the fairness of the experiment, we make the following setup for all comparison algorithms. The mean job arrival rate is fixed at 20, furthermore, the proportion of secure VMs and computing intensive jobs both account for 50%.

As shown in Figure 6, regardless of the proportion of private cloud VMs, PH-DRL outperforms the comparison algorithms in terms of success rate and average response time. Moreover, it is apparent to find that, along with the growth of the proportion of private cloud VMs, the success rate increases and the response time decreases, which indicates that PH-DRL performs better when more private cloud VMs are in a hybrid cloud.

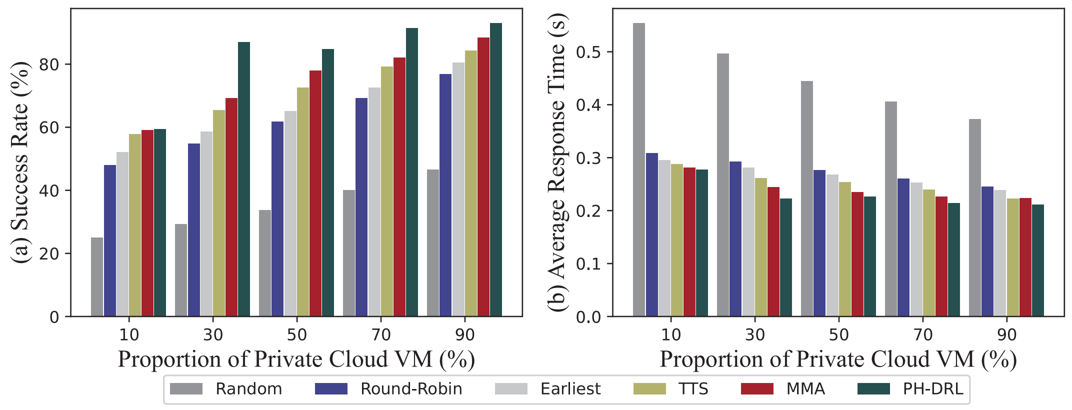

FIGURE 6 | Performance under different private cloud VM proportion.

6 | Conclusion

Job scheduling in hybrid clouds poses a significant challenge for cloud-based application providers. In this paper, we have proposed a job scheduling framework called PH-DRL, which has taken into account privacy and QoS requirements. Specifically, we have leveraged the real-time decision-making capabilities of DRL and have developed an intelligent and efficient scheduling approach that prioritizes privacy preservation in hybrid cloud systems while ensuring optimal performance. We have provided a comprehensive description of our method, and our experimental results have convincingly demonstrated that the proposed approach exhibits superior performance compared to existing methods. Furthermore, PH-DRL exhibits excellent scalability, as it can easily adapt to user's evolving requirements and environmental changes with minimal modifications. In our future work, we plan to further develop the method to handle more complex workloads in cloud, such as workflows from end users, with privacy preservation in mind.

Acknowledgments

Part of this work was supported by the Fundamental Research Funds for the Central Universities (2023YQ002).

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. F. Fang and X. Wu, "A Win–Win Mode: The Complementary and Coexistence of 5G Networks and Edge Computing," *IEEE Internet of Things Journal* 8, no. 6 (2020): 3983–4003.

2. T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications* (Perth, WA: IEEE, 2010), 27–33.

3. J. Liu, H. Shen, H. Chi, et al., "A Low-Cost Multi-Failure Resilient Replication Scheme for High-Data Availability in Cloud Storage," *IEEE/ACM Transactions on Networking* 29, no. 4 (2020): 1436–1451.

4. M. Armbrust, A. Fox, R. Griffith, et al., "A View of Cloud Computing," *Communications of the ACM* 53, no. 4 (2010): 50–58.

5. E. Deelman, "Grids and Clouds: Making Workflow Applications Work in Heterogeneous Distributed Environments," *International Journal of High Performance Computing Applications* 24, no. 3 (2010): 284–298.

6. Y. Wei, L. Pan, S. Liu, L. Wu, and X. Meng, "DRL-Scheduling: An Intelligent QoS-Aware Job Scheduling Framework for Applications in Clouds," *IEEE Access* 6 (2018): 55112–55125.

7. Z. Wen, R. Qasha, Z. Li, R. Ranjan, P. Watson, and A. Romanovsky, "Dynamically Partitioning Workflow Over Federated Clouds for Optimising the Monetary Cost and Handling Run-Time Failures," *IEEE Transactions on Cloud Computing* 8, no. 4 (2016): 1093–1107.

8. J. Zhu, X. Li, R. Ruiz, and X. Xu, "Scheduling Stochastic Multi-Stage Jobs to Elastic Hybrid Cloud Resources," *IEEE Transactions on Parallel and Distributed Systems* 29, no. 6 (2018): 1401–1415.

9. Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao, and J. Chen, "Scheduling Workflows With Privacy Protection Constraints for Big Data Applications on Cloud," *Future Generation Computer Systems* 108 (2020): 1084–1091.

10. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine* 34, no. 6 (2017): 26–38.

11. S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced Multi-Verse Optimizer for Task Scheduling in Cloud Computing Environments," *Expert Systems With Applications* 168 (2021): 114230.

12. M. Abd Elaziz and I. Attiya, "An Improved Henry Gas Solubility Optimization Algorithm for Task Scheduling in Cloud Computing," *Artificial Intelligence Review* 54, no. 5 (2021): 3599–3637.

13. B. M. H. Zade, N. Mansouri, and M. M. Javidi, "SAEA: A Security-Aware and Energy-Aware Task Scheduling Strategy by Parallel Squirrel Search Algorithm in Cloud Environment," *Expert Systems With Applications* 176 (2021): 114915.

14. L. F. Bittencourt, E. R. Madeira, and N. L. Da Fonseca, "Scheduling in Hybrid Clouds," *IEEE Communications Magazine* 50, no. 9 (2012): 42–47.

15. H. Yuan, J. Bi, and M. Zhou, "Temporal Task Scheduling of Multiple Delay-Constrained Applications in Green Hybrid Cloud," *IEEE Transactions on Services Computing* 14, no. 5 (2018): 1558–1570.

16. Q. H. Zhu, H. Tang, J. J. Huang, and Y. Hou, "Task Scheduling for Multi-Cloud Computing Subject to Security and Reliability Constraints," *IEEE/CAA Journal of Automatica Sinica* 8, no. 4 (2021): 848–865.

17. Q. Liu, L. Cheng, A. L. Jia, and C. Liu, "Deep Reinforcement Learning for Communication Flow Control in Wireless Mesh Networks," *IEEE Network* 35, no. 2 (2021): 112–119.

18. P. R. Wurman, S. Barrett, K. Kawamoto, et al., "Outracing Champion Gran Turismo Drivers With Deep Reinforcement Learning," *Nature* 602, no. 7896 (2022): 223–228.

19. Q. Liu, T. Xia, L. Cheng, M. Van Eijk, T. Ozcelebi, and Y. Mao, "Deep Reinforcement Learning for Load-Balancing Aware Network Control in IoT Edge Systems," *IEEE Transactions on Parallel and Distributed Systems* 33, no. 6 (2021): 1491–1502.

20. J. James, W. Yu, and J. Gu, "Online Vehicle Routing With Neural Combinatorial Optimization and Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems* 20, no. 10 (2019): 3806–3817.

21. B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV Trajectory Planning in Wireless Sensor Networks for Energy Consumption Minimization by Deep Reinforcement Learning," *IEEE Transactions on Vehicular Technology* 70, no. 9 (2021): 9540–9554.

22. Y. Huang, L. Cheng, L. Xue, et al., "Deep Adversarial Imitation Reinforcement Learning for QoS-Aware Cloud Job Scheduling," *IEEE Systems Journal* 16, no. 3 (2022): 4232–4242.

23. Y. Gu, F. Cheng, L. Yang, J. Xu, X. Chen, and L. Cheng, "Cost-Aware Cloud Workflow Scheduling Using DRL and Simulated Annealing," *Digital Communications and Networks* (2024), https://doi.org/10.1016/j.dcan.2023.12.009.

24. L. Cheng, Y. Wang, F. Cheng, C. Liu, Z. Zhao, and Y. Wang, "A Deep Reinforcement Learning-Based Preemptive Approach for Cost-Aware Cloud Job Scheduling," *IEEE Transactions on Sustainable Computing* 9, no. 3 (2023): 422–432.

25. H. Baniata, A. Anaqreh, and A. Kertesz, "PF-BTS: A Privacy-Aware Fog-Enhanced Blockchain-Assisted Task Scheduling," *Information Processing and Management* 58, no. 1 (2021): 102393.

26. J. Lei, Q. Wu, and J. Xu, "Privacy and Security-Aware Workflow Scheduling in a Hybrid Cloud," *Future Generation Computer Systems* 131 (2022): 269–278.

27. S. Sharif, P. Watson, J. Taheri, S. Nepal, and A. Y. Zomaya, "Privacy-Aware Scheduling SaaS in High Performance Computing Environments," *IEEE Transactions on Parallel and Distributed Systems* 28, no. 4 (2016): 1176–1188.

28. H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for Workflows With Security-Sensitive Intermediate Data by Selective Tasks Duplication in Clouds," *IEEE Transactions on Parallel and Distributed Systems* 28, no. 9 (2017): 2674–2688.

29. Y. Zhang, J. Sun, and Z. Wu, "An Heuristic for Bag-of-Tasks Scheduling Problems With Resource Demands and Budget Constraints to Minimize Makespan on Hybrid Clouds," in *2017 Fifth International Conference on Advanced Cloud and Big Data* (Shanghai, China: IEEE, 2017), 39–44.

30. R. Xie, D. Gu, Q. Tang, T. Huang, and F. R. Yu, "Workflow Scheduling in Serverless Edge Computing for the Industrial Internet of Things: A Learning Approach," *IEEE Transactions on Industrial Informatics* 19, no. 7 (2023): 8242–8252.

31. S. Jin, S. Wang, and F. Fang, "Game Theoretical Analysis on Capacity Configuration for Microgrid Based on Multi-Agent System," *International Journal of Electrical Power & Energy Systems* 125 (2021): 106485.

32. L. Cheng, Y. Gu, Q. Liu, L. Yang, C. Liu, and Y. Wang, "Advancements in Accelerating Deep Neural Network Inference on Aiot Devices: A Survey," *IEEE Transactions on Sustainable Computing* (2024). https://doi.org/10.1109/TSUSC.2024.3353176

33. V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Playing Atari With Deep Reinforcement Learning," arXiv Preprint arXiv:1312.5602 2013.

34. V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature* 518, no. 7540 (2015): 529–533.