山西大学学报(自然科学版)48(6):1080—1091,2025
Journal of Shanxi University(Nat. Sci. Ed.)

DOI:10.13451/j.sxu.ns.2025016

基于图神经网络的云桌面虚拟机调度算法

彭商濂[1],柳岸[2*]
(1.成都信息工程大学 计算机学院,四川 成都 610225;
2.四川川大智胜软件股份有限公司,四川 成都 610045)

摘 要:随着云桌面系统的广泛应用,虚拟机调度算法面临着如何有效应对复杂多变负载的挑战。传统算法在资源利用率、系统延迟和负载均衡等方面表现欠佳。本文提出一种基于图神经网络与强化学习相结合的云桌面虚拟机调度优化算法。将云桌面环境中的虚拟机及其资源需求建模为图结构,利用图神经网络预测虚拟机的负载情况。结合强化学习策略,根据预测结果动态调整资源分配和虚拟机迁移决策,优化系统性能。使用4K视频处理、办公应用、网络应用等场景的多种数据集,评估算法的资源利用率、系统延迟和负载均衡等指标。实验结果表明,基于图神经网络与强化学习的调度算法在多个数据集上均表现出优异的性能,在资源利用率上提升了12%以上,系统延迟减少了15%,负载均衡效果优于常用资源调度算法。

关键词:虚拟机调度;图神经网络;强化学习;资源优化;动态负载分配
中图分类号:TP39 **文献标志码**:A **文章编号**:0253-2395(2025)06-1080-12

Cloud Desktop Virtual Machine Scheduling Algorithm Based on Graph Neural Networks

PENG Shanglian[1], LIU An[2*]
(1. *College of Computer, Chengdu University of Information Technology, Chengdu* 610225*, China;*
2. *Wisesoft Co., Ltd., Chengdu* 610045*, China*)

Abstract: With the widespread application of cloud desktop systems, virtual machine scheduling algorithms face the challenge of effectively handling complex and dynamic workloads. Traditional scheduling algorithms often underperform in terms of resource utilization, system latency, and load balancing. This paper proposes a cloud desktop virtual machine scheduling optimization algorithm that combines graph neural networks (GNN) and reinforcement learning (RL). We model the virtual machines and their resource requirements in the cloud desktop environment as a graph structure, and use GNN to predict the load conditions of the virtual machines. By incorporating RL strategies, we dynamically adjust resource allocation and virtual machine migration decisions based on the prediction results to optimize system performance. The algorithm is evaluated on multiple datasets from real-world scenarios, including 4K video processing, office applications, and network applications, by measuring indicators like resource utilization, system latency, and load balancing. Experimental results show that the proposed scheduling algorithm exhibits significant improvements across multiple datasets. Compared to traditional algorithms, resource utilization increases by over 12%, system latency is reduced by 15%, and load balancing is significantly better.

Key words: virtual machine scheduling; graph neural networks; reinforcement learning; resource optimization; dynamic load balancing

收稿日期:2025-01-15;**接受日期**:2025-03-08
基金项目:四川省科技厅重点研发项目(2023YFG0144)
作者简介:彭商濂(1980—),男,四川遂宁人,博士,讲师,研究方向为数据管理,知识图谱。E-mail:psl@cuit.edu.cn
* **通信作者**:柳岸(LIU An),E-mail:1215544@qq.com
引文格式:彭商濂,柳岸.基于图神经网络的云桌面虚拟机调度算法[J].山西大学学报(自然科学版),2025,48(6):1080-1091. DOI:10.13451/j.sxu.ns.2025016.

0 引言

随着云计算和虚拟化技术的快速发展,云桌面系统作为一种新兴的IT基础设施解决方案,正广泛应用于企业和个人用户中。在云桌面环境中,虚拟机的调度优化问题成为了系统性能提升的关键。传统的虚拟机调度算法大多依赖于预定义的规则和简单的启发式方法,这些方法在面对多变且复杂的负载时,表现出了明显的局限性。虚拟机调度不仅需要考虑资源的有效利用,还需要应对系统延迟和负载均衡等多重挑战[1]。因此,如何提高调度算法的智能化水平,已成为云桌面虚拟机调度领域的一个研究热点[2]。

传统的基于负载均衡的云桌面虚拟机调度算法包括最小负载算法(Least Load)[3]、最小连接数算法(Least Connection)[4]、加权轮询算法(Weighted Round Robin)[5]等。这些调度算法在某些特定场景下能够取得一定的效果,但它们普遍存在以下几个问题:(1)资源利用率不高,导致系统的整体效能未能得到充分发挥;(2)系统延迟较高,尤其是在高负载情况下,可能导致用户体验不佳;(3)负载分配不均衡,部分虚拟机可能面临资源不足,而其他虚拟机则可能处于空闲状态,浪费了计算资源。这些问题严重影响了云桌面环境的可扩展性和响应能力。

在近年来的虚拟机资源调度领域,许多研究集中在提升系统性能、负载均衡和容错能力等方面[6-16]。为了优化云计算环境中的资源分配,研究者们提出了多种方法,其中最为广泛关注的是基于机器学习、深度学习和图神经网络(Graph Neural Network, GNN)[17]的优化策略。

GNN作为一种强大的图结构建模工具,已被广泛应用于多种资源调度问题中。通过对云计算资源之间的关系进行建模,GNN能够有效捕捉资源调度中的复杂依赖关系,从而优化负载分配和故障预测。此外,结合深度学习等技术,研究者们提出了多种基于智能算法的调度方法,如深度强化学习(Reinforcement Learning, RL)、遗传算法(Genetic Algorithm, GA)等,以期通过动态调整和优化策略来提高系统的性能和鲁棒性。

文献[18]提出了一种基于机器学习的优化方法,利用机器学习算法预测用户需求,并根据预测结果动态调整资源分配策略,从而减少资源浪费并提升系统响应速度。

对云计算环境中动态任务调度的鲁棒性和截止时间保障问题,文献[19]提出了一种基于元深度强化学习(Meta Deep Reinforcement Learning, MDRL)的调度解决方案。该方法通过量化鲁棒性指标(如重新训练时间),在高度动态的任务负载和资源可用性变化下,提升了调度性能的鲁棒性和适应速度。

文献[20]提出一种基于注意力时空卷积的虚拟机主动容错优先级迁移决策模型,利用带有注意力机制的长短期记忆网络提取每个主机的时间特征,并结合多主机之间的交互信息构建图网络,使用图注意力网络提取网络中不同主机的关联信息,并利用这些信息编码主机的故障信息,该模型在故障检测、能耗和延迟敏感性方面优于现有基准方法。

文献[21]提出了一种结合图神经网络与动态多队列优化调度的方法,用于提升云计算环境中的故障容忍度、负载均衡性和系统性能。该方法利用GNN对云计算资源进行建模,预测可能的故障并优化资源调度决策;然后,通过动态调整多队列优化调度系统,根据云负载的变化,实时调整资源分配策略,以实现负载均衡和提高系统效率。

文献[22]提出了一种混合深度学习模型(Discrete Particle Swarm Optimization and Genetic Algorithm, DPSO-GA),用于云计算中的动态工作负载调度。该方法结合了粒子群优化(Particle Swarm Optimization, PSO)和GA,通过两阶段的优化过程提升资源利用率和负载均衡效果。在第一阶段,PSO和GA相结合的优化策略用于超参数调优,解决了工作负载预测中的挑战。第二阶段采用卷积神经网络(Convolutional Neural Network, CNN)与长短期记忆网络(Long short-term memory, LSTM)相结合的深度学习模型,预测云资源的未来需求。通过CNN的特征提取能力和LSTM的时间序列预测能力,该模型能够有效捕捉工作负载的动态变化。

文献[23]提出了一种基于层次强化学习的调度算法(Multi-dimensional Hierarchical Reinforcement Learning, MD-HRL),通过高层代理

(H-Agent)和低层代理(L-Agent)两级架构优化异构计算平台中的任务调度和资源分配。H-Agent结合多跳注意力图神经网络和一维卷积神经网络编码任务和资源信息,并通过Kolmogorov-Arnold网络计算优先级;L-Agent则利用双深度Q网络优化任务与资源的映射。

文献[24]提出了一种基于图神经网络和深度强化学习的层次协作资源调度框架,以解决工业物联网中边缘计算环境下的复杂调度问题。该框架利用层次图神经网络促进层内节点与相邻节点之间的无缝信息交换,并将其转化为节点嵌入表示。然后将嵌入信息输入策略模型中,通过迭代学习过程,利用全局特征信息生成高质量的调度方案。

尽管已有虚拟机资源调度研究在传统云计算领域取得了良好的成果,但针对云桌面虚拟机调度的研究仍然较少。现有的基于GNN的调度方法主要集中在静态或较为简单的场景,且大多数方法未能有效考虑动态变化的工作负载和资源状况。此外,现有的调度算法往往缺乏对多维度性能优化(如延迟、资源利用率、系统可靠性等)的综合考虑,这在实际应用中可能会限制其有效性。

本文研究基于图神经网络的云桌面虚拟机调度优化算法,通过利用GNN对虚拟机负载进行预测和动态调整,以实现资源的高效分配和利用。

本文的主要贡献包括如下几个方面:

1)提出一种基于图神经网络的云桌面虚拟机调度优化算法。通过建模虚拟机及其资源需求为图结构,并结合GNN对虚拟机负载进行预测,实现了对复杂多变负载的有效处理,提升了调度算法的效率和性能。

2)利用强化学习策略,结合预测结果进行动态资源调整和虚拟机迁移决策,实现了资源的高效分配和利用。相较于传统调度算法,该算法在资源利用率、系统延迟等方面均取得了显著提升,为云桌面系统的性能优化带来了新的思路和方法。

3)通过实验验证,基于GNN的调度算法在不同数据集下均展现出了优异的表现,有效地提升了云桌面系统的整体性能和用户体验,为虚拟机调度优化领域带来了新的技术突破和应用前景。

1 基于图神经网络的虚拟机资源调度模型

在本节中,我们主要介绍基于图神经网络的虚拟机资源调度模型。

1.1 基本模型描述

假设对于一个给定云桌面系统,其中包含N个虚拟机(Virtual Machine, VM),每个VM可以表示为图中的一个节点v_i。这些节点可以形成一个虚拟机集合的图$G=(V,E)$,其中$V=\{v_1,v_2,\cdots,v_n\}$是节点集合,E是边集合,用于表示VM之间的关系。

对于每个节点v_i,可以定义节点的特征向量$\boldsymbol{x}_i$,其中包括了该虚拟机的资源需求、运行状态等信息。例如,假设每个节点的特征向量包含CPU使用率、内存需求等方面的信息,则可以表示为$X_i=[x_{i1},x_{i2},\cdots,x_{im}]$。此外,我们定义一个邻接矩阵$\boldsymbol{A}$来表示图中节点之间的关系。如果节点$v_i$和$v_j$之间存在边,则$A_{ij}=1$,否则$A_{ij}=0$。

图1为本文提出的基于图神经网络和强化学习的虚拟机调度模型的整体结构。模型由四个主要部分组成:输入层、GNN层1、GNN层2以及输出层。输入层包含多个虚拟机节点(如VM1、VM2、VM3),每个节点携带其关键资源特征,如CPU使用率和内存需求。GNN层1通过图卷积操作实现消息传递,初步整合虚拟机之间的资源依赖关系;GNN层2进一步进行特征聚合,深化特征表示,以捕捉更复杂的系统状态信息。最终,输出层基于GNN层2生成的丰富特征,结合强化学习策略,做出具体的资源分配和虚拟机迁移决策,实现高效的资源利用和负载均衡。

整个信息流动过程中,输入层的虚拟机节点通过GNN层1和GNN层2的多层图卷积操作,不断更新和优化节点特征,使其能够准确反映系统的动态负载状态。通过这种方式,模型不仅能够高效提取和整合虚拟机之间的复杂关系,还能在输出层生成智能化的调度决策,优化资源分配,降低系统延迟。该结构结合了GNN在图数据处理中的优势与RL在动态决策优化中的能力,具备良好的可扩展性和鲁棒性,能够适应不同规模和多变环境下的云桌面系统需求,显著提升系统的整体性能和用户体验。

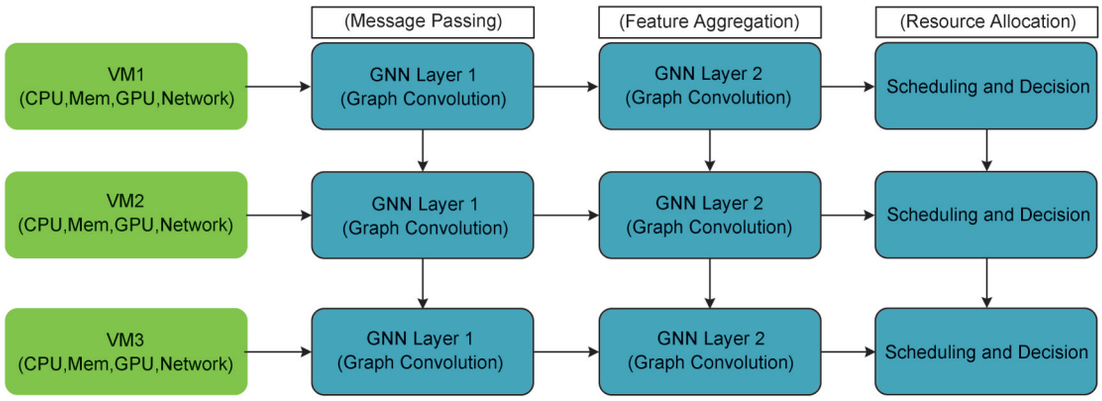

图1 基于图神经网络和强化学习的虚拟机调度模型

Fig. 1 Virtual machine scheduling model based on graph neural networks and reinforcement learning

1.2 图神经网络构建

现在,我们设计图神经网络模型$f(\boldsymbol{X},\boldsymbol{A};\Theta)$,其中$\boldsymbol{X}$表示所有节点的特征向量构成的矩阵,$\boldsymbol{A}$表示邻接矩阵,$\Theta$表示模型的参数集合。

模型f的具体结构包括以下几个组件:

(1)节点特征更新函数

$$H^{(l+1)}=\phi(H^{(l)},\ \boldsymbol{A};\Theta^{(l)}),\tag{1}$$

其中$H^{(l)}$表示第l层节点的隐藏表示,$\Theta^{(l)}$表示第l层的参数。

(2)图结构学习函数ρ

$$Z=\rho(H^{(l)},\ \boldsymbol{A};\Theta^{(l)}),\tag{2}$$

其中Z表示整个图的表示,$\Theta^{(l)}$表示最后一层的参数。

(3)输出预测函数g

$$Y=g(Z,\ \boldsymbol{A};\Theta^{(\mathrm{out})}),\tag{3}$$

其中Y表示最终的输出,可用于最佳的虚拟机调度策略。

在设计这些组件时,需要考虑如何利用节点特征更新函数ϕ对每个节点的特征进行更新,如何利用图结构学习函数ρ对整个图的特征进行学习,以及如何利用输出预测函数g对最终的调度结果进行预测。

该模型帮助我们建立一个完整的图神经网络模型,结合虚拟机调度问题进行节点特征的学习和图结构的表示,可以实现更有效的虚拟机调度和资源管理。

1.3 GNN模型参数优化

下面详细介绍如何结合虚拟机调度任务,通过反向传播算法和优化器对GNN模型的参数进行优化。GNN模型参数优化流程如下:

(1)前向传播:计算预测值

在前向传播过程中,我们输入虚拟机及其资源需求构成的图$G=(V,E)$,以及对应的节点特征矩阵$\boldsymbol{X}$和邻接矩阵$\boldsymbol{A}$到GNN模型中,得到预测结果。前向传播计算预测值算法如算法1所示。

算法1 前向传播计算预测值算法

输入:

$\boldsymbol{X}$:节点特征矩阵,维度为$(N,\ F)$,其中N是节点数量,F是特征维度;

$\boldsymbol{A}$:邻接矩阵,维度为$(N,\ N)$;

W:图神经网络的层权重列表,每个元素维度为$(F_l\times F_{l+1})$;

多层感知机的参数;

激活函数σ(ReLU);

L:图神经网络的层数;

输出:预测值矩阵Z(维度:$N\times$C,其中C是预测类别或回归目标的数量)。

① $\hat{A}=\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ /* 计算归一化的邻接矩阵,其中$\boldsymbol{D}$是度矩阵*/

② $H^{(0)}=\boldsymbol{X}$ /*初始化节点特征*/

③ For l=1 to L: /*图卷积层的前向传播*/

$H^{(l)}=\sigma(\hat{A}H^{(l-1)}W^{(l-1)})$;/*计算第l层的节点特征*/

④ End For

⑤ Z=MLP($H^{(L)}$) /*使用MLP对最后一层的节点特征进行映射*/

⑥ return Z /*返回预测值*/

在算法1中,层权重矩阵$\boldsymbol{W}$的维度由每层的输入特征维度F_l和输出特征维度F_{l+1}决定。具体来说,输入特征维度F_l是由前一层的输出特征维度F_{l-1}决定的,并且在设计网络结构时可以根据具体任务和数据特征进行调整。例如,对于初始输入层,输入特征维度F_0由节点的初始特征决定,如CPU使用率、内存需求等。在隐藏层,特征维度通常通过实验调整以优化模型性能,常见的做法是逐层增加或减少特征维度,以提取更高级的特征。最终输出层的特征维度F_l则根据具体任务的输出需求决定,例如分类任务的类别数或回归任务的目标变量数。因此,特征维度F_l的选择是一个设计和调优的过程,需要结合具体应用和实验结果进行确定。

(2)损失函数计算算法

损失函数用于衡量模型预测结果与实际结果之间的差异,是模型训练过程中优化的目标。在本文中,损失函数用于衡量模型预测结果与实际结果(模型训练过程中实际观察到的或通过仿真环境获取的真实数据)之间的差异。对于GNN模型,实际结果是指虚拟机在给定时间点的真实负载需求,这些负载数据来源于历史调度记录和实际系统监控的数据,反映了虚拟机的实际资源使用情况。具体而言,这些实际负载数据通常是通过系统监控工具(如Prometheus、Nagios等)或在实验环境中进行负载模拟得到的。对于强化学习任务,实际结果则来自于通过仿真环境进行资源调度后的系统性能反馈,包括资源利用率、系统延迟、负载均衡等。我们通过比较模型预测的虚拟机负载和调度决策与实际结果来优化模型的性能。

GNN在云桌面虚拟机调度任务中的损失函数如算法2所示。

算法2 损失函数计算算法

输入:

预测值矩阵$\boldsymbol{Z}$(维度:$N\times C$,其中C是预测类别或回归目标的数量)

实际值矩阵$\boldsymbol{Y}_i$(维度:$N\times C$);

损失函数类型;

输出:损失值$\mathcal{L}$。

① 根据任务要求(回归或分类)选择合适的损失函数

② IF regression /*回归任务损失函数*/

$$\mathcal{L}=\frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{Y}_i-\boldsymbol{Z}_i)^2$$

③ IF classification/*分类任务损失函数*/

$$\mathcal{L}=-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\boldsymbol{Y}_{ic}\log(\boldsymbol{Z}_{ic})$$

④ return $\mathcal{L}$ /*返回损失值*/

(3)反向传播:计算梯度

通过反向传播算法,计算损失函数相对于模型参数的梯度。具体步骤包括:

① 计算损失函数对每层输出的偏导数。

② 使用链式法则逐层向后计算梯度,直到输入层。

③ 对每个参数$\theta\in\Theta$,计算其梯度$\frac{\partial\mathcal{L}}{\partial\theta}$。

(4)优化器更新参数

使用优化器根据计算出的梯度更新模型参数,本文采用随机梯度下降(Stochastic Gradient Descent, SGD):

$$\Theta^{(t+1)}=\Theta^{(t)}-\eta\frac{\partial\mathcal{L}}{\partial\theta}。\tag{4}$$

2 强化学习策略结合GNN的虚拟机调度算法

在云桌面虚拟机调度中,单独依赖GNN仍然存在一些问题和局限性,这些问题主要体现在以下几个方面:

(1)局部性问题:GNN的消息传递机制依赖于节点邻居的信息,因此,它在处理具有高度复杂和动态变化的全局负载时,可能无法完全捕捉到系统的全局状态。这会导致虚拟机的负载预测不够准确,尤其是在负载变化迅速或复杂的场景中。

(2)实时性问题:虚拟机调度对实时性有较高的要求,GNN在训练和预测过程中可能会消耗大量的计算资源和时间,这可能会影响调度的及时性和响应速度。

(3)动态性问题:云桌面系统的虚拟机负载是动态变化的,而GNN模型通常在静态图结构上进行训练和预测。虽然GNN可以进行多次迭代来处理动态变化的数据,但这种方法在面对频繁变化的负载时,可能会显得不够高效。

(4)迁移决策问题:GNN擅长预测和分

类,但在具体的资源调整和虚拟机迁移决策方面,其能力有限。虚拟机迁移决策不仅需要考虑当前的负载状态,还需要考虑未来的负载变化、网络条件、资源限制等复杂因素。

为了克服上述GNN的局限性,我们引入了强化学习策略,结合GNN的预测结果进行动态资源调整和虚拟机迁移决策。这样可以利用RL在动态决策和全局优化方面的优势,实现资源的高效分配和利用。

为了训练图神经网络模型,我们首先从云桌面系统中收集虚拟机的资源使用数据,包括CPU使用率、内存需求、I/O负载等。此外,还需要知道虚拟机所在物理主机的资源情况,如CPU核心数、内存大小、存储资源、网络带宽等。数据收集后,我们进行数据预处理,包括归一化和标准化处理,以确保不同虚拟机的资源需求能够在同一尺度下进行处理。特征向量构建包括虚拟机的当前资源需求、物理主机的资源状态和虚拟机与物理主机之间的拓扑关系等信息。负载标签则来自于历史调度记录和模拟实验,反映虚拟机的实际资源需求,并作为训练目标。通过历史数据,我们生成训练集和验证集,确保GNN模型能够学习时间序列数据中的负载变化模式,提升负载预测的准确性。

强化学习模型的训练数据集主要由虚拟机的资源状态、系统负载信息、资源配置和网络状态等组成。状态空间包括虚拟机的CPU使用率、内存需求、I/O负载等信息,以及当前的资源配置和网络状况。通过在仿真环境中进行训练,RL模型根据当前的系统状态进行动作选择,这些动作包括调整虚拟机资源分配(如CPU、内存)和虚拟机迁移决策。为了生成训练数据,我们使用历史调度数据和模拟实验来构建系统的状态和动作空间,并基于任务的系统性能(如资源利用率、延迟、负载均衡等)设计奖励函数。通过与仿真环境的交互,RL模型能够优化虚拟机调度策略,实现资源的动态分配和负载均衡。

基于强化学习策略结合GNN的虚拟机调度算法过程如下:

(1) GNN负载预测

首先,通过GNN对云桌面环境中的虚拟机及其资源需求进行建模和预测。具体步骤如下:

输入数据:构建包含虚拟机和物理主机的图结构,其中节点表示虚拟机和主机,边表示它们之间的关系。

消息传递:利用GNN的消息传递机制,从邻居节点中聚合信息,更新每个节点的特征表示。

预测输出:通过多层感知机或其他适当的输出层,对未来一段时间的虚拟机负载进行预测。

其数学表示如下:

$$h_i^{(l+1)}=\sigma\left(W^{(l)}\sum_{j\in N(i)}\frac{1}{c_{ij}}h_j^{(l)}+b^{(l)}\right),\qquad(5)$$

其中$h_i^{(l)}$是节点i在第l层的特征表示,$N(i)$是节点i的邻居集合,c_{ij}是归一化系数,$W^{(l)}$和$b^{(l)}$是可训练的参数,σ是激活函数。

(2) 状态空间定义

将虚拟机的负载预测值、当前资源配置、网络状态等信息作为强化学习的状态输入。状态空间S定义为:$S=\{\text{Predicted Load, Current Resource Configuration, Network Status}\}$,即状态空间包括虚拟机的CPU使用率、内存使用率、磁盘I/O、网络延迟等。

(3) 动作空间定义

动作空间A包括以下操作:$A=\{\text{Increase CPU, Decrease CPU, Increase Memory, Decrease Memory, Migrate VM}\}$。这些动作涉及调整虚拟机的资源分配(增加或减少CPU、内存)以及将虚拟机从一台物理主机迁移到另一台。

(4) 奖励函数设计

设计奖励函数R,根据资源利用率、系统延迟、负载均衡等指标给予奖励或惩罚。奖励函数的目标是最大化资源利用率,最小化系统延迟,实现负载均衡,奖励函数R如下所示:

$$R=\alpha\times U-\beta\times L-\gamma\times I,\qquad(6)$$

其中α、β、γ是权重参数,用于平衡各项指标,U为资源利用率,L为系统延迟,I为负载不均衡度。

(5) 策略优化

本文利用Q-learning对策略网络进行优化,使其能够在不同负载场景下做出最优的资源调整和迁移决策。具体过程包括:

①策略更新:使用Bellman方程更新Q值:

$$Q(s,\ a)\leftarrow Q(s,\ a)+\alpha(r+\gamma\max_{a'}Q(s',\ a')-Q(s,\ a)),\tag{7}$$

其中α是学习率,γ是折扣因子,r是即时奖励,s'是执行动作后的新状态。

②策略改进:基于最新的Q值,选择在当前状态下收益最大的动作。

$$\pi(s)=\arg\max_a Q(s,\ a)。\tag{8}$$

(6)动态资源调整和迁移决策

根据RL策略网络的输出,实时调整虚拟机的资源配置和迁移策略。具体步骤包括:

①资源调整:根据策略网络的决策,动态增加或减少虚拟机的CPU和内存资源。

②迁移决策:在需要时,将虚拟机从负载高的主机迁移到负载低的主机,以实现负载均衡。

3 实验分析

3.1 实验环境描述

为验证基于图神经网络和强化学习相结合的虚拟机调度算法在云桌面环境中的性能,我们搭建了如下的实验平台。实验环境包括多台物理主机,每台主机上运行若干虚拟机。具体实验设置如下:

(1)硬件环境

物理主机数量:10台。

每台主机配置:16核CPU、64 GB内存、1 TB硬盘。

虚拟机数量:每台主机上运行10~20个虚拟机。

(2)软件环境

操作系统:Ubuntu 20.04。

虚拟化平台:KVM。

图神经网络框架:PyTorch Geometric。

强化学习框架:Stable Baselines3。

(3)数据集

使用Stress-ng[25]工具模拟云桌面虚拟机的负载情况,包括CPU负载、内存负载和I/O负载等。我们通过设置不同的参数,模拟超高清视频融媒体平台的场景,支持4K视频格式,包括3 840×2 160 4K UHD、4 096×2 160 4K和4 096×2 304分辨率。这些视频格式需要高度的资源支持和处理能力,在平台上的负载具有高度的变动性和波动性,需要及时的资源调整和负载均衡来保证视频播放和处理的顺畅性和稳定性。

本文中,图神经网络和强化学习的训练数据集的组织如下:GNN训练数据集的输入特征包括虚拟机和物理主机的资源使用情况(如CPU使用率、内存需求、带宽等)以及网络拓扑信息,标签为期望的调度结果,这些数据通过历史调度记录和模拟实验生成。RL训练数据集的状态表示当前系统的资源使用状态,动作为调度策略,奖励基于调度结果的性能指标计算,数据集通过RL代理与模拟环境交互生成,模拟环境反映了真实云平台的运行情况。为确保训练过程中的数据准确性并避免不匹配的结果,我们在数据集构建过程中进行数据清洗和验证。历史调度记录经过预处理和校验,确保虚拟机负载标签的真实有效。同时,强化学习模型的训练在真实云平台的仿真环境中进行,该环境高度还原了实际系统的运行情况。在每次模拟实验后,对实验结果进行回溯分析,以确保每个训练样本和结果的匹配性,从而避免训练数据的误导。

(4)对比算法

为全面评估算法性能,我们实现了如下算法的对比:传统负载均衡算法,包括轮询算法(Round Robin,RR)、最短作业优先算法(Shortest Job First,SJF)和先来先服务算法(First Come First Served,FCFS);基于GA启发式虚拟机调度,以及基于机器学习随机森林(Random Forest,RF)的虚拟机调度方法。

3.2 实验结果

(1)资源利用率

在模拟的4K视频数据集上,采用传统的算法和本文的算法对云桌面系统的CPU、内存和I/O的负载进行了测试,其资源利用率对比如图2所示。

从图2可以看出,本文算法相对于传统调度算法在资源利用率上表现出较大的优势,平均提升了12%以上的资源利用率。该算法在网络应用和图形渲染等高资源要求场景下表现突出,同时对系统性能稳定性和延迟的优化效果也十分明显。

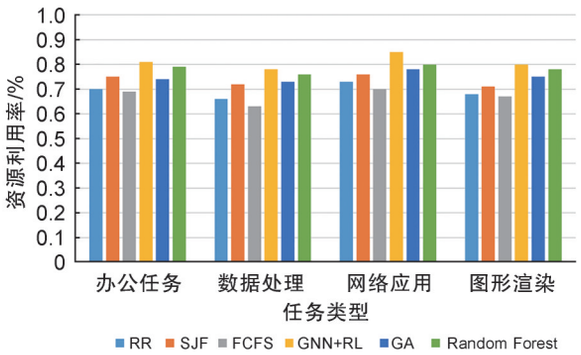

图2 不同负载下算法的资源利用率对比

Fig. 2 Resource utilization comparison of algorithms under different workloads

(2)系统延迟

我们模拟了四类不同的任务用于系统延迟的测试。测试结果如图3所示。

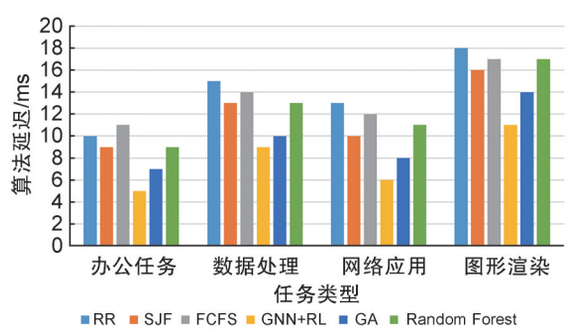

图3 不同任务下算法的延迟对比

Fig. 3 Delay comparison of algorithms under different tasks

从图3可以看出,基于GNN+RL的调度算法在系统延迟方面表现出色,通过有效的资源分配和虚拟机迁移策略,实现了更短的任务处理时间和更高的任务处理效率。相比之下,传统负载均衡算法(如轮询算法、最小连接数算法)在处理任务时延迟较高,容易出现任务等待时间过长的情况,导致系统响应速度较慢。启发式算法和基于机器学习的虚拟机调度算法在系统延迟方面介于传统算法和GNN+RL算法之间,部分场景下可能出现较长的任务处理时间。

(3)负载均衡

负载均衡测试主要评估各算法在不同负载条件下的资源分配和负载均衡效果。几种算法在不同任务下的负载均衡对比如图4所示。

从图4可以看出,基于GNN+RL的调度算法在负载均衡方面表现出色,通过有效的资源分配和虚拟机迁移策略,避免了单一主机负载过高的情况,实现了更加平衡的负载分布。相比之下,传统负载均衡算法(如轮询算法、最小连接数算法)在处理负载不均衡方面效果较弱,容易出现某些主机过载的情况,导致资源浪费和系统性能下降。启发式算法(如遗传算法)和基于机器学习的随机森林虚拟机调度算法在负载均衡方面表现介于传统算法和GNN+RL算法之间,部分场景下可能出现负载不均衡的情况。因此,GNN+RL算法在实现负载均衡方面具有明显的优势。

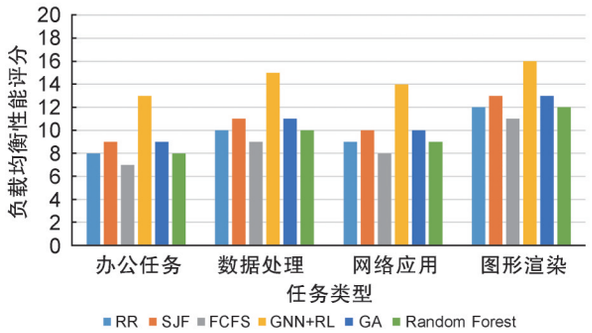

图4 不同任务下算法的负载均衡性能评分对比

Fig. 4 Load balancing score comparison of algorithms under different tasks

(4)虚拟机迁移次数

虚拟机迁移次数指标反映了虚拟化环境中虚拟机的迁移频率和效率。虚拟机迁移次数通常与负载均衡、资源利用率和系统稳定性密切相关。几种算法的虚拟机迁移次数对比如图5所示。

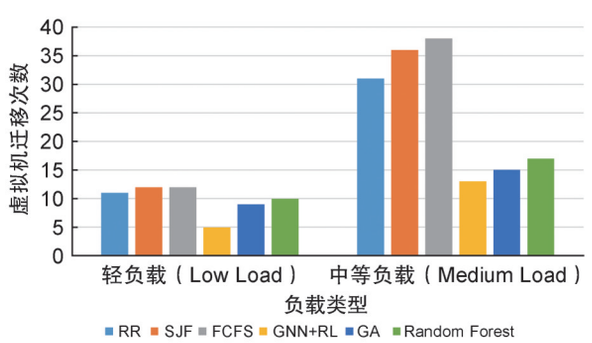

图5 不同任务下算法的虚拟机迁移次数对比

Fig. 5 VM migration times comparison of algorithms under different tasks

图5实验结果显示,相比于传统负载均衡算法和启发式算法,基于机器学习的虚拟机调度和本文算法在轻负载和中等负载下能够显著减少虚拟机迁移次数,降低了系统资源开销和性能损耗。特别是本文算法,在模拟超高清视

频融媒体平台场景下,相对传统算法和启发式算法,虚拟机迁移次数减少了20%~25%,表明其在负载管理和资源调度方面的优越性。

(5)GNN模型泛化能力

为了验证所提出的GNN模型在不同云桌面负载场景下的泛化能力,本实验旨在评估模型在未见过的负载类型和不同系统规模下的预测性能。具体而言,通过在多种典型负载场景(如办公应用、网络应用、视频处理和科学计算)中训练和测试GNN模型,探讨其在跨场景负载预测中的适应性和稳定性。此外,实验还将考察GNN模型在虚拟机数量和物理主机规模变化时,能否保持高效的资源负载预测能力,从而验证其在实际大规模云桌面系统中的应用潜力。

本实验设计涵盖多类型负载数据集和不同规模的云桌面系统,以全面评估GNN模型的泛化性能。首先,构建四种典型负载场景的数据集:办公应用负载(场景A)、网络应用负载(场景B)、视频处理负载(场景C)和科学计算负载(场景D)。每种场景分别在小规模(5台物理主机,每台5~10个虚拟机)、中规模(10台物理主机,每台10~20个虚拟机)和大规模(20台物理主机,每台20~30个虚拟机)的系统配置下进行训练和测试。GNN模型首先在单一负载场景(例如场景A)上进行训练,然后在其他未见过的负载场景(场景B、C、D)及不同规模系统上进行测试,记录预测准确率(如均方误差MSE)和模型稳定性指标。此外,通过引入正则化技术和调整网络结构,进一步优化模型的泛化能力。实验结果将通过表格和图示形式展示,比较不同场景和规模下GNN模型的预测性能,以验证其在多样化和动态变化环境中的适应性。

图6为GNN模型在四种不同负载场景(办公应用、网络应用、视频处理、科学计算)及三种系统规模(小规模、中规模、大规模)下的均方误差(Mean Squared Error, MSE),并与传统机器学习方法(随机森林,Random Forest)和基线模型(线性回归,Linear Regression)进行了对比。

从图6的实验结果可以看到,本文的图神经网络(GNN)模型在所有负载类型和系统规模下均优于传统的机器学习方法。具体而言,GNN模型在小规模系统下分别实现了最低的均方误差(MSE)0.045和0.035,显示出其在捕捉和预测复杂资源需求方面的卓越能力。随着系统规模从小规模扩展到大规模,GNN模型的MSE仅略有上升(办公应用从0.045增加到0.060,视频处理从0.035增加到0.045),表明其具备良好的可扩展性和鲁棒性。在负载波动较大的网络应用和科学计算场景中,GNN模型同样保持了显著低于随机森林和线性回归的MSE值,提升幅度在10%至30%之间。这一贯穿不同负载类型和系统规模的优异表现,充分证明了GNN模型出色的泛化能力及其在动态复杂云桌面环境中的有效性。此外,随着系统规模的扩大,预测误差的增长幅度较小,进一步说明GNN模型适用于大规模部署,是提升资源分配与负载均衡在实际云桌面应用中的可靠选择。

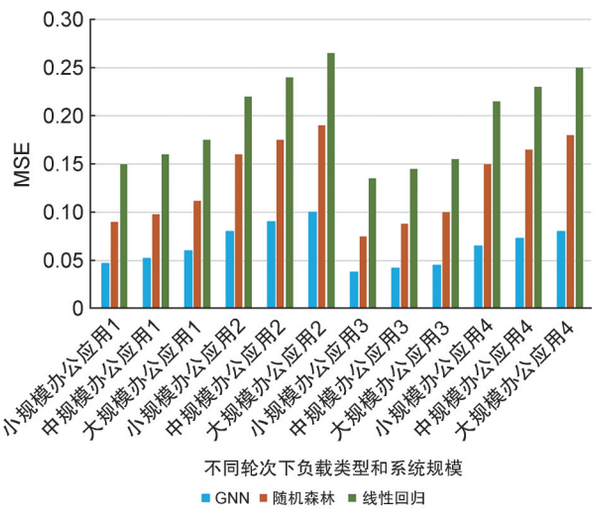

图6 GNN模型在不同负载类型和系统规模下的MSE对比

Fig. 6 Comparison of GNN model's prediction performance across different load types and system scales

(6)强化学习策略的参数敏感性

为了深入探讨强化学习策略中关键超参数对虚拟机调度性能的影响,并验证所提出的基于图神经网络与强化学习相结合的调度算法在不同参数设置下的鲁棒性与适应性,本实验通过系统评估学习率(α)、折扣因子(γ)以及探索策略(ϵ)等核心超参数对资源利用率、系统延迟和负载均衡效果的具体影响。此外,实验还将通过引入网络延迟抖动和虚拟机故障等环境干扰因素,考察强化学习策略在复杂和动态变化的云桌面环境中的稳定性和鲁棒性,以确保其在实际应用中的可靠性和高效性。

本实验设计涵盖了不同学习率($\alpha=1\times10^{-3}$、5×10^{-4}、1×10^{-4})、折扣因子($\gamma=0.80$、0.90、0.95、0.99)以及初始探索率($\epsilon=0.2$、0.3、0.4)的多种组合,通过系统调节这些超参数,观察其对强化学习策略收敛速度、资源利用率、系统延迟和迁移次数等关键性能指标的影响。实验过程中,部分实验将模拟10%的网络延迟抖动和5%的虚拟机故障,以测试强化学习策略在存在环境干扰时的鲁棒性。同时,采用均方误差(MSE)、资源利用率(Resource Utilization)、系统延迟(System Latency)和负载均衡度(Load Balancing)等指标,对不同参数配置下的策略性能进行全面评估和对比分析,以确定最优参数组合并验证RL策略在实际云桌面系统中的应用潜力。

表1展示了不同学习率(α)和折扣因子(γ)组合下,强化学习策略在收敛后的资源利用率和系统延迟表现。

表1 不同学习率与折扣因子组合下的调度表现

Table 1 Scheduling performance under different combinations of learning rates and discount factors

α	γ	收敛迭代数	资源利用率/%	系统延迟/ms	迁移次数	波动方差
1×10^{-3}	0.80	900	85.2	61	120	较大
1×10^{-3}	0.90	950	86.9	59	115	略大
5×10^{-4}	0.90	1 200	90.3	52	100	中等
5×10^{-4}	0.95	1 300	90.8	51	98	中等
1×10^{-4}	0.90	2 000+	88.5	53	105	较小
1×10^{-4}	0.95	2 200+	89.2	50	102	小

从表1可以看出,较高的学习率(1×10^{-3})加快了策略的收敛速度,但在后期容易导致策略不稳定,表现为系统延迟的较大波动。较低的学习率(1×10^{-4})虽然提高了策略的稳定性,但收敛速度显著下降。

较高的折扣因子(如0.95)使得策略更加注重长期收益,能够有效降低系统延迟,但可能导致资源利用率的提升略受限制。较低的折扣因子(如0.80)则偏向短期优化,虽能快速提升资源利用率,但系统延迟控制效果不如高折扣因子。

最优组合:实验结果表明,学习率设定为5×10^{-4},折扣因子在0.90至0.95之间时,能够在资源利用率、系统延迟和迁移次数之间达到较好的平衡。

我们也对探索策略(ϵ-greedy)的影响进行了实验。实验中初始探索率设定为0.2、0.3、0.4,观察其对策略收敛和系统性能的影响。表2显示了不同初始探索率下的资源利用率和系统延迟。

表2 不同学习率与折扣因子组合下的调度表现

Table 2 Scheduling performance under different combinations of learning rates and discount factors

初始ϵ	收敛迭代数	资源利用率/%	系统延迟/ms	迁移次数
0.2	1 300	90.3	52	100
0.3	1 250	90.1	53	102
0.4	1 350	90.5	51	98

从表2可以看出,较高的初始探索率(0.4)在策略训练初期促进了更广泛的策略探索,最终资源利用率略有提升且系统延迟较低。然而,过高的探索率可能延长收敛时间。初始探索率为0.2和0.3时,策略收敛较快,且系统性能接近最优状态。

对于收敛迭代数,在初始探索率较高时,收敛迭代数有所增加,但最终系统性能有所提升,表明适度的探索能够帮助强化学习策略找到更优的调度策略。

4 结论

本文针对云桌面系统中虚拟机调度算法的优化问题,提出了一种基于图神经网络(GNN)和强化学习(RL)相结合的虚拟机调度优化算法。该算法通过GNN对虚拟机负载进行准确预测,并结合RL策略进行动态资源调整和虚拟机迁移决策,实现了资源的高效分配与利用。实验结果表明,基于GNN与RL的调度算法在资源利用率、系统延迟等关键指标上均优于传统调度算法,证明了其在复杂多变的云桌面环境中的有效性和优越性。

尽管本文所提出的方法在虚拟机调度优化方面取得了显著的成果,但仍存在一些局限性,未来有进一步改进的空间。

(1)GNN模型的动态适应性与实时性

当前的GNN模型在处理高度动态的全局负载时,可能难以实时捕捉系统状态的快速变化,影响调度决策的及时性。未来可以通过引

入动态图神经网络(Dynamic Graph Neural Networks)或增量学习技术,使模型能够实时更新图结构和节点特征,从而更迅速地适应负载变化。此外,优化GNN的计算效率,例如采用更轻量级的图卷积操作或并行计算技术,可以进一步提升模型的实时预测能力,确保调度决策的及时性和准确性。

(2)强化学习策略的决策复杂性

目前的RL策略在进行资源调整和虚拟机迁移决策时,主要依赖于当前负载预测结果,可能忽视了一些长远的系统状态和潜在的资源瓶颈。未来可以探索多目标强化学习(Multi-objective Reinforcement Learning)或层次化强化学习(Hierarchical Reinforcement Learning),以同时优化多个性能指标,如资源利用率、系统延迟和能耗等。此外,结合深度强化学习(Deep Reinforcement Learning)中的先进算法,如Proximal Policy Optimization(PPO)或Deep Q-Network(DQN)变体,可以提升策略网络的决策质量和稳定性。

(3)模型泛化能力与参数调优

虽然GNN模型在不同负载类型和系统规模下表现出良好的泛化能力,但在实际应用中,面对更多样化和复杂的负载模式,仍需进一步提升模型的泛化性能。未来可以通过迁移学习(Transfer Learning)或元学习(Meta-Learning)技术,增强GNN模型在不同场景下的适应能力。同时,采用自动化的超参数优化方法,如贝叶斯优化(Bayesian Optimization),可以有效地提升模型的预测精度和鲁棒性,减少人工调参的工作量。

(4)系统可扩展性与容错机制

在大规模云桌面系统中,调度算法需要具备良好的可扩展性和容错能力。未来研究可以结合分布式计算框架,如Apache Spark或Flink,将GNN与RL算法分布式部署,以提升算法的处理能力和扩展性。同时,设计有效的容错机制,如虚拟机快照和自动恢复策略,可以提高系统在面对硬件故障或网络异常时的稳定性和可靠性。

通过上述改进措施,基于GNN和RL的虚拟机调度算法将在动态适应性、决策质量、模型泛化能力以及系统可扩展性等方面得到显著提升,进一步增强其在实际云桌面系统中的应用效果和性能表现。这些未来的研究方向不仅能够克服现有算法的局限性,还将为云桌面系统的资源管理和性能优化提供更加高效和智能的解决方案。

参考文献:

[1] MANSOUR J, GIORDANI J, MORESI L, *et al*. Underworld2: Python Geodynamics Modelling for Desktop, HPC and Cloud[J]. *J Open Source Softw*, 2020, **5**(47): 1797. DOI:10.21105/joss.01797.

[2] 方义秋, 唐道红, 葛君伟. 云环境下基于虚拟机动态迁移的调度策略研究[J]. 微电子学与计算机, 2012, **29**(4): 45-48. DOI: 10.19304/j.cnki.issn1000-7180.2012.04.011.
FANG Y Q, TANG D H, GE J W. Research on Schedule Strategy Based on Dynamic Migration of Virtual Machines in Cloud Environment[J]. *Microelectron Comput*, 2012, **29**(4): 45-48. DOI: 10.19304/j. cnki. issn1000-7180.2012.04.011.

[3] AZAR Y. On-line Load Balancing[M]//Online Algorithms. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998: 178-195. DOI:10.1007/bfb0029569.

[4] SINGH G, KAUR K. An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems[J]. *Int Res J Eng Technol*, 2018, **5**(3): 6.

[5] 张英静, 何锋, 卢广山, 等. 基于TTE的改进加权轮询调度算法[J]. 北京航空航天大学学报, 2017, **43**(8): 1577-1584. DOI: 10.13700/j.bh.1001-5965.2016.0590.
ZHANG Y J, HE F, LU G S, *et al*. A Modified Weighted round Robin Scheduling Algorithm in TTE[J]. *J Beijing Univ Aeronaut Astronaut*, 2017, **43**(8): 1577-1584. DOI: 10.13700/j.bh.1001-5965.2016.0590.

[6] XU H, XU S, WEI W, *et al*. Fault Tolerance and Quality of Service Aware Virtual Machine Scheduling Algorithm in Cloud Data Centers[J]. *J Supercomput*, 2023, **79**(3): 2603-2625. DOI:10.1007/s11227-022-04760-5..

[7] 张哲宇, 周润林, 孙斌, 等. XMPP协议的数据分发网络的负载均衡算法[J]. 北京邮电大学学报, 2016, **39**(S1): 27-31. DOI: 10.13190/j.jbupt.2016.s.007.
ZHANG Z Y, ZHOU R L, SUN B, *et al*. Load Balancing Algorithm for Data Distribution Network Based on XMPP Protocol[J]. *J Beijing Univ Posts Telecommun*, 2016, **39**(S1): 27-31. DOI: 10.13190/j.jbupt.2016.s.007.

[8] 闫兴亚, 韩萍. 云计算环境中负载均衡技术的研究与应用[J]. 微电子学与计算机, 2015, **32**(10): 181-184. DOI: 10.19304/j.cnki.issn1000-7180.2015.10.039.

YAN X Y, HAN P. Research and Application of Load Balance Technology in Cloud Computing Environment [J]. *Microelectron Comput*, 2015, **32**(10): 181−184. DOI: 10.19304/j.cnki.issn1000-7180.2015.10.039.

[9] CHITRA DEVI D, RHYMEND UTHARIARAJ V. Load Balancing in Cloud Computing Environment Using Improved Weighted round Robin Algorithm for Nonpreemptive Dependent Tasks[J]. *Sci World J*, 2016, **2016**: 3896065. DOI:10.1155/2016/3896065.

[10] 徐胜超. 贪心算法优化云数据中心的虚拟机分配策略[J]. 计算机系统应用, 2021, **30**(3): 134−141. DOI: 10.15888/j.cnki.csa.007814.
XU S C. Greedy Algorithms Optimized Virtual Machine Allocation for Cloud Data Centers[J]. *Comput Syst Appl*, 2021, **30**(3): 134−141. DOI: 10.15888/j.cnki.csa.007814.

[11] WU G, TANG M L, TIAN Y C, *et al*. Energy-efficient Virtual Machine Placement in Data Centers by Genetic Algorithm[M]//Neural Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012: 315−323. DOI:10.1007/978-3-642-34487-9_39.

[12] TSENG F H, CHEN X J, CHOU L D, *et al*. Support Vector Machine Approach for Virtual Machine Migration in Cloud Data Center[J]. *Multimed Tools Appl*, 2015, **74**(10): 3419−3440. DOI:10.1007/s11042-014-2086-z.

[13] KUMAR K V, MALATHI P, RAMESH S. Performance Analysis of Placement Prediction System Using Support Vector Machine over Random Forest Algorithm[C]// 2022 14th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS). New York: IEEE, 2022: 1−4. DOI:10.1109/MACS56771.2022.10023271.

[14] RAMAN N, WAHAB A B, CHANDRASEKARAN S. Computation of Workflow Scheduling Using Backpropagation Neural Network in Cloud Computing: A Virtual Machine Placement Approach[J]. *J Supercomput*, 2021, **77**(9): 9454−9473. DOI:10.1007/s11227-021-03648-0.

[15] HUMMAIDA A R, PATON N W, SAKELLARIOU R. Scalable Virtual Machine Migration Using Reinforcement Learning[J]. *J Grid Comput*, 2022, **20**(2): 15. DOI:10.1007/s10723-022-09603-4. E Communications Letters, 2020, **25**(1): 176−180. DOI: 10.1109/LCOMM.2020.3025298.

[16] SUN P H, LAN J L, LI J F, *et al*. Combining Deep Reinforcement Learning with Graph Neural Networks for Optimal VNF Placement[J]. *IEEE Commun Lett*, 2020, **25**(1): 176−180. DOI:10.1109/LCOMM.2020.3025298.

[17] SCARSELLI F, GORI M, TSOI A C, *et al*. The Graph Neural Network Model[J]. *IEEE Trans Neural Netw*, 2009, **20**(1): 61−80. DOI:10.1109/TNN.2008.2005605.

[18] ZHANG Y F, LIU B, GONG Y L, *et al*. Application of Machine Learning Optimization in Cloud Computing Resource Scheduling and Management[C]//Proceedings of the 5th International Conference on Computer Information and Big Data Application. New York: Association for Computing Machinery, 2024.

[19] LIU H Y, CHEN P, OUYANG X, *et al*. Robustness Challenges in Reinforcement Learning Based Time-critical Cloud Resource Scheduling: A Meta-learning Based Solution[J]. *Future Gener Comput Syst*, 2023, **146**: 18−33. DOI:10.1016/j.future.2023.03.029.

[20] 党伟超, 武婷玉. 基于注意力时空卷积和A2C的虚拟机主动容错优先迁移决策[J]. 计算机应用研究, 2023, **40**(12): 3606−3613. DOI: 10.19734/j. issn. 1001-3695.2023.03.0144.
DANG W C, WU T Y. Active Fault-tolerant Priority Migration Decision Model for Virtual Machines Based on Attentional Spatio-temporal Convolution and A2C [J]. *Appl Res Comput*, 2023, **40**(12): 3606−3613. DOI: 10.19734/j.issn.1001-3695.2023.03.0144.

[21] KALASKAR C, THANGAM S. A Graph Neural Network-based Approach with Dynamic Multiqueue Optimization Scheduling (DMQOS) for Efficient Fault Tolerance and Load Balancing in Cloud Computing[J]. *Int J Intell Syst*, 2024, **2024**(1): 6378720. DOI:10.1155/int/6378720.

[22] SIMAIYA S, LILHORE U K, SHARMA Y K, *et al*. A Hybrid Cloud Load Balancing and Host Utilization Prediction Method Using Deep Learning and Optimization Techniques[J]. *Sci Rep*, 2024, **14**(1): 1337. DOI: 10.1038/s41598-024-51466-0.

[23] ZHANG Z, XU C, LIU K, *et al*. A Resource Optimization Scheduling Model and Algorithm for Heterogeneous Computing Clusters Based on GNN and RL[J]. *J Supercomput*, 2024, **80**(16): 24138−24172. DOI:10.1007/s11227-024-06383-4.

[24] MENG Q F, LUO Z H, ZHENG X L, *et al*. Hierarchical Collaborative Resource Scheduling in Industrial Internet of Things Based on Graph Neural Networks and Deep Reinforcement Learning[C]//2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS). New York: IEEE, 2023: 2647−2654. DOI:10.1109/ICPADS60453.2023.00351.

[25] KING C I. Stress-ng (Version 0.17.09)[CP/OL]. (2024-04-16) [2024-12-01]. https://github. com/ColinIanKing/stress-ng.