基于强化学习的算力网络管理和调度

王 静[1),2)] 谢 鲲[1),2)] 曾 鑫[1),2)] 赵鹏程[1),2)] 文吉刚[3)] 谢高岗[4)]

[1)](湖南大学信息科学与工程学院 长沙 410082)

[2)](超算与人工智能融合计算教育部重点实验室 长沙 410082)

[3)](湖南科技大学计算机科学与工程学院 湖南 湘潭 411100)

[4)](中国科学院计算机网络信息中心 北京 100190)

摘 要 算力网络作为一种新兴的网络架构,通过整合分散的计算资源,构建一个统一的资源池,实现计算能力的动态按需分配。然而,现有的调度策略在算力资源和网络资源分配之间解耦合,无法为时延敏感型任务提供稳定的时延保障。针对这一挑战,本文提出了一种基于强化学习的算力网络联合调度优化框架(RLMS-CPN),通过实时监测网络状态和算力资源分布,智能地进行算力节点选择与路径规划,旨在最小化任务的响应时延。该框架包含多项创新技术,包括面向开放网络环境的AI算力任务计算时延估计模型,以及基于图神经网络的强化学习算网联合调度算法。通过在NS3模拟系统中对算力网络控制系统进行仿真,我们在四种不同的网络拓扑下进行实验。结果表明,与现有方法相比,RLMS-CPN在降低响应时延方面取得了显著成效,平均降低了22.61%的响应时延,显著提升了网络整体性能和用户体验。此外,针对新设备的计算时延估计,两阶段微调时延估计模型(TSFT)在仅使用50个新样本的情况下,实现了0.003的低预测误差。

关键词 算力网络;时延敏感型任务;流量工程;任务卸载;强化学习

中图法分类号 TP393 **DOI号** 10.11897/SP.J.1016.2025.02713

Reinforcement Learning-Based Management and Scheduling in Computing Power Networks

WANG Jing[1),2)] XIE Kun[1),2)] ZENG Xin[1),2)] ZHAO Peng-Cheng[1),2)]
WEN Ji-Gang[3)] XIE Gao-Gang[4)]

[1)](College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082)

[2)](Key Laboratory of Supercomputing and Artificial Intelligence Fusion Computing, Ministry of Education, Changsha 410082)

[3)](School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan 411100)

[4)](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

Abstract As an innovative network architecture, the computing power network (CPN) integrates distributed computing resources to form a unified, scalable resource pool, enabling users to dynamically access computing power on demand and achieve efficient resource utilization. Its core concept is to treat computing power as a service, allowing users to flexibly access it without needing to concern themselves with the specific configuration of underlying resources. Although computing power networks offer significant flexibility and scalability, existing technologies still face challenges when addressing new latency-sensitive tasks (such as virtual reality, autonomous

收稿日期:2025-01-08;在线发布日期:2025-07-22。本课题得到国家自然科学基金杰出青年基金(No. 62025201)、国家自然科学基金委员会与香港研究资助局合作研究重点项目(No. 62321166652)资助。**王 静**,硕士研究生,主要研究方向为计算机网络,算力网络。E-mail: wangjingwr@hnu.edu.cn。**谢 鲲(通信作者)**,博士,教授,中国计算机学会(CCF)高级会员,主要研究方向为计算机网络、网络安全、算力网络、人工智能。E-mail:xiekun@hnu.edu.cn。**曾 鑫**,博士研究生,主要研究方向为网络测量、带内遥测。**赵鹏程**,硕士研究生,主要研究方向为网络安全、算力网络。**文吉刚**,博士,副教授,主要研究方向为网络安全、算力网络。**谢高岗**,博士,研究员,中国计算机学会(CCF)高级会员,主要研究方向为网络体系结构、网络测量。

driving, and telemedicine). Such tasks require computing and transmission processes to be completed within strict latency constraints to ensure service quality and user experience. Task response latency is composed of computational latency and transmission latency, where computational latency depends on the performance of computing nodes, while transmission latency is influenced by both the selection of computing nodes and path planning. Therefore, it is necessary to achieve global latency minimization through the coordinated optimization of computational and network resource scheduling. However, existing computing network scheduling research mostly inherits from the fields of cloud computing and edge computing, focusing solely on single-resource optimization, such as traffic engineering (optimizing transmission paths) or task offloading (selecting computing nodes), and lacks joint optimization of computing node selection and path planning. This loosely coupled scheduling approach struggles to simultaneously reduce computational and transmission latency, leading to insufficient overall task execution efficiency. Additionally, the heterogeneity of computing nodes and the dynamism of network paths further increase the complexity of collaborative scheduling. To address the shortcomings of existing computing networks in latency-sensitive task scheduling, this paper proposes a reinforcement learning-based computing network scheduling and control framework (RLMS-CPN) that deeply integrates computing and transmission resources to achieve intelligent collaborative optimization of node selection and path planning. This framework includes three core innovations: First, a reinforcement learning-based joint scheduling optimization mechanism is designed, which dynamically adjusts resource allocation by real-time monitoring of network status and node computing power information. Experiments validate that it reduces task response latency by an average of 22.61% across four datasets; Second, to address the latency prediction challenges posed by new devices joining heterogeneous environments, a two-stage fine-tuning strategy (TSFT) based on neural networks is proposed. This strategy uses a small amount of labeled data for model fine-tuning and enhances the training effect of unlabeled data through pseudo-labeling technology, ultimately achieving an average prediction accuracy of 0.003 on new devices; Finally, we innovatively designed the SyncGNN graph neural network structure, which unifies the modeling of computational and network resource characteristics through feature fusion technology. Compared to traditional MLP methods, this approach improves latency performance by 18.75% and effectively solves the decoupling problem between node selection and path planning. These innovations collectively form a comprehensive intelligent scheduling solution for computing power networks, significantly enhancing the processing efficiency of latency-sensitive tasks. The system proposed in this paper achieves the first intelligent collaborative scheduling of computing power and network resources for AI tasks, providing efficient services for latency-sensitive tasks by jointly optimizing computational and transmission latency.

Keywords computing power network; latency-sensitive tasks; traffic engineering; task offloading; reinforcement learning

1 引 言

1.1 背景及动机

算力网络作为一种新兴的网络架构,通过将计算资源整合于网络之中,构建出一个统一的、可扩展的资源池。这种架构使得用户能够根据需求动态获取和分配计算资源,实现资源的高效利用。算力网络的核心思想是将计算能力视为一种服务,用户可以像使用水电一样按需使用计算资源,无需关心资源的具体位置和配置[1-4]。

尽管算力网络展现出了灵活性和高效性,但现有技术难以完全满足新型任务的需求,尤其随着越来越多的时延敏感型任务涌现,这些任务要求在严

格的时延范围内完成计算与传输,以确保服务质量和用户体验。例如:虚拟现实技术[5]、自动驾驶[6]、远程医疗[7]。

时延敏感型任务的执行流程主要包括两个关键阶段:传输和计算,其响应时延由这两部分时延组合而成。计算时延主要由算力任务和算力节点的性能决定;而传输时延则不仅取决于算力节点的选择,还受到传输路径规划的影响。在算力网络中,为了满足时延敏感型任务的高算力和低时延需求,必须协同优化计算资源和网络资源的调度,以在计算和传输两个方面同时降低时延。

然而,专注于算力网络调度策略的研究较少[8-11],现有调度策略主要继承于云计算和边缘计算,仅关注单一资源的优化,如流量工程[12-30]和任务卸载[31-37]。流量工程主要集中于任务传输方式的优化,而任务卸载则侧重于将计算任务分配到合适的计算节点。具体而言,流量工程仅考虑路径规划,而未充分考虑算力节点的选择;相反,任务卸载虽然重视算力节点的选择,但忽视了路径规划的问题。这种松耦合的调度方法无法有效协调算力节点的选择与路径规划,导致无法同时减少计算时延和网络传输时延,从而增加了任务的总体执行时延,难以满足时延敏感性任务的需求。

在算力网络中,分布式算力节点具有不同硬件配置与计算能力,这对任务的计算时延产生直接影响。同时,从算力任务接入点到算力节点之间存在多条传输路径,这些路径的物理特性、带宽容量和拥塞状况等因素,会显著影响数据传输效率,进而影响传输时延。因此,在任务执行时,为了实现整体的资源分配,必须综合考虑各个算力节点的计算能力和路径的传输效率,这些都为协同调度带来挑战。

1.2　贡　献

针对时延敏感型任务对低时延的迫切需求,本文提出了一种基于强化学习的算力网络管理与调度框架(RLMS-CPN)。该框架通过紧密耦合计算资源与传输资源,实现了智能选择计算节点和流量调度分配的优化,从而有效地满足时延敏感型任务的需求。

本文的具体贡献总结如下:

(1)基于强化学习的算力网络联合调度优化框架

针对算力网络中计算资源和网络资源分配的解耦合问题,提出了一种基于强化学习的算力网络联合调度优化框架。该框架能够实现算力节点选择与路径规划的协同优化,以最小化任务响应时延。通过实时检测网络状态和算力节点信息,算力网络控制器能够做出智能决策,提升资源调度效率。在四个数据集上的广泛实验评估表明,与现有方法相比,RLMS-CPN 在降低响应时延方面具有显著优势,平均降低了 22.61% 的响应时延。

(2)面向开放空间下的计算时延估计模型

为了准确估计 AI 任务在异构算力节点的计算时延,提出一种基于神经网络的计算时延预测模型。该模型旨在解决开放空间,新设备算力节点不断加入时,现有估计模型无法适配新设备上的任务时延估计的问题。为此,本文设计了一种两阶段微调策略(TSFT),以提高开放算力网络环境中对新设备计算时延的预测准确性。在第一阶段,利用少量有标签样本对神经网络进行微调;在第二阶段,借助微调后的模型为未标记样本生成伪标签,再次训练模型。这一策略使得模型展现出对新设备的高适应能力,在多种新设备上实现了平均 0.003 的预测误差。

(3)面向节点选择和路径规划联合优化的图神经网络(Graph Neural Network, GNN)结构

针对算力网络中节点选择与路径规划之间松耦合的问题,设计了一种新的 GNN 结构(SyncGNN)。该结构有效整合计算资源和网络传输资源,实现了特征融合。通过 SyncGNN,能够协同优化算力节点的选择和路径规划,提高资源调度效率,最大限度地减少任务响应时延,并提升整体系统性能。与传统的多层感知器(Multi-Layer Perceptron,简称 MLP)相比,SyncGNN 在时延性能上提升了 18.75%,显示出其在算力网络资源调度特征融合中的显著优势。

综上所述,本文提出的算力网络控制系统通过综合优化任务的计算时延和传输时延,实现了算力及网络资源的合理分配和高效利用,为时延敏感型任务提供更高效的网络服务。据我们所知,本文是首个面向 AI 任务的算力资源及网络资源协同调度系统,专注于利用智能算法实现资源的联合优化,以满足算力任务的低响应时延要求。

2　相关工作

本论文的研究背景扎根于算力网络,深度关联流量工程与任务卸载这两大核心领域。流量工程主要关注路径规划,影响任务的传输时延;任务卸载则关注算力节点的选择,影响任务的计算时延和传输

时延。此外,为了实现协同调度,计算时延预测也是关键环节。

2.1 算力网络资源调度

算力网络资源调度,作为算力网络的核心问题,旨在对网络中的算力资源、网络资源等进行合理分配,以满足各类应用任务对计算能力与数据传输的需求。它需要综合考虑网络拓扑结构、算力性能以及任务的特性,通过优化调度算法,保障任务高效、稳定地执行,进而提升整个算力网络的服务质量与运行效益。

文献[8]提出了一种面向算力感知网络的多维度资源分配机制,结合 GNN 和深度强化学习,增强了网络拓扑的泛化能力。文献[9]设计了基于 Deep Q-Learning(简称 DQN)的智能流量调度策略,通过节点计算负载预测优化路由选择,降低了丢包率和流完成时间,提升了算网融合系统的调度性能。文献[10]开发了基于 Q-Learning 的软件定义算力路由算法,利用强化学习智能体和软件定义网络(Software-Defined Networking,简称 SDN)的全局视图,实现了算力和网络资源的智能管理和协同调度,为任务调度和数据传输提供了高效解决方案。文献[11]提出了一种面向多维协同的算网融合光网络路由与资源分配算法,通过计算节点资源均衡度和合适度选择最优中间节点进行计算卸载,并根据路径合适度选择承载业务的路径,优化了资源分配,降低了业务阻塞率,提高了频谱利用率。

这些研究工作为算力网络的高效调度提供了理论支持和技术方法,但是他们没有针对 AI 任务,同时对计算时延和传输时延进行协同优化。

2.2 流量工程

流量工程是网络优化的关键领域,其核心目标是通过精心设计的路由策略,提升网络的整体性能和效率。解决方案主要分为两大类方法:

链路级流量工程:通过解决多商品流问题[12],利用网络中的任意链路进行传输,旨在最小化最大链路利用率。

路径级流量工程:在预设路径上进行传输,相较于链路级流量工程,具有更低的复杂度。

在网络中实施有效的路由策略,可以降低网络中的最大链路利用率,有效缓解拥塞问题。通常,这些问题被构建为线性规划模型,输入流量矩阵,并使用求解器或启发式算法来寻找最优解。

2.2.1 线性规划与启发式

线性规划方法(如 Gurobi 求解器[13])能够为网络流量需求提供全局最优解,但其计算复杂度随网络规模剧增。为此,有研究者提出启发式算法,利用线性规划求解器分配最高的 $k\%$ 流量需求,剩余需求按照最短路径路由[14]。当前流量工程主要基于开放最短路径优先(Open Shortest Path First,简称 OSPF)等路由协议来优化网络性能。这些协议通过调整链路权重而不是修改路由协议本身来进行路由规划。尽管这种方法在静态流量模式下效果显著,但在处理突发流量和网络异常时可能无法及时适应,限制了其在现代动态网络环境中的性能。加权等价成本多路径[15]是对等价成本多路径的扩展,允许进行更精细的流量分割。

但是,基于线性规划或者启发式算法的流量工程解决方案无法对时延进行建模,因为它们无法将时延表述成一个线性表达式。现有研究尝试采用 Markovian Arrival Process / Markovian Service Process / Single Server 队列模型来近似传输时延,但实际网络时延的复杂性远超单一队列模型的表征能力。

2.2.2 基于强化学习的流量工程

随着人工智能技术的飞速发展,强化学习在流量工程领域展现出了巨大的潜力。文献[16]提出基于 GNN 的快速流量工程方案,通过构建端到端可微分的网络模型,利用梯度下降动态调整 OSPF 链路权重。文献[17]提出基于深度强化学习的动态流量工程框架,通过动态调整网络中关键链路的虚拟容量属性重构网络,选择性重路由 Top-k 关键流。该方法有效缓解了重路由带来的计算开销,但可扩展性仍需验证。文献[18]将 Transformer 模型与深度强化学习结合,通过 Transformer 模型捕捉网络状态时序特征,并利用深度强化学习建立连续状态空间与路由策略之间的直接映射,从而有效解决动态流量需求下的流量工程问题。该模型虽提升了时间序列建模能力,但参数规模随网络规模增长,限制了其扩展性。文献[19-20]通过动态调整关键链路权重生成路由策略,但关键节点的选择策略缺乏理论支撑。文献[21-24]结合线性规划和强化学习优势,智能筛选关键流量进行重路由,其余流量采用默认路由,在效率与效果间取得了较好平衡,但线性规划求解器仍可能成为性能瓶颈。

尽管上述方法取得一定进展,但在网络规模扩展时面临状态空间爆炸与计算开销难题,为此,学术界开始探索多智能体强化学习的分布式解决方案。

文献[25]提出了新的框架,融合 GNN、多智能体强化学习和交替方向乘子法,提升大规模广域网流量工程的计算效率,但多智能体之间缺乏协同。文献[26]提出了一种基于 GNN 和多智能体强化学习的分布式流量工程框架,优化 OSPF 协议的链路权重,以最小化网络拥塞。文献[27]针对动态混合软件定义网络环境,提出了一种基于多智能体强化学习的分布式流量工程方法。该方法通过构建合理的虚拟环境来训练路由智能体,并引入差异奖励分配机制以协调多个智能体共同优化全局流量工程目标。文献[28]通过在每个 SDN 节点上部署一个智能体,仅利用本地信息来做出路由决策,从而最小化网络的全局最大链路利用率。文献[29]通过结合层次化控制平面和多智能体深度强化学习,设计了一种可扩展且高效的流量工程解决方案,能够在多域网络中优化路由决策,同时保护隐私并满足服务质量要求。文献[30]通过在每个网络节点部署多个智能体来独立决策流量路由,并引入联合训练技术以加速学习过程并提高系统性能。

尽管当前的流量工程解决方案在网络路由优化方面取得了一定的进展,但是针对算力网络的资源调度问题,既需要考虑计算资源,又需要考虑网络资源。目前没有工作实现面向 AI 任务的算力资源及网络资源协同调度系统。

2.3　任务卸载

任务卸载是云计算和边缘计算领域的一项关键技术,其核心目标是为计算任务选择合适的算力执行节点,以实现计算任务的优化执行。

文献[31]提出协作式两阶段优化框架,通过凸优化技术解耦任务分割与资源分配实现系统效用的最大化。文献[32]提出混合卸载算法,旨在减少延迟并增强边缘环境中的无服务器计算性能。文献[33]利用深度强化学习来优化卸载决策,旨在减少延迟并提高用户体验。文献[34]建立边缘计算分布式优化模型,提出了一个用户关联和资源分配方案,目标最小化服务延迟。文献[35]通过联合优化任务卸载决策和计算资源分配,最小化多接入边缘计算系统中的时间和能耗成本。文献[36]通过建立包含本地边缘计算资源、宏基站和子基站的边缘服务器资源以及云计算服务器资源的卸载模型,将优化问题分解为资源分配和卸载策略两个子问题交替求解,实现了系统的延迟和能耗最小化。文献[37]提出了一种基于多目标的联合优化算法,通过构建包含本地计算资源、边缘服务器资源和通信模型的系统模型,实现延迟和能耗的联合优化。

尽管当前任务卸载研究在资源调度方面取得了进展,但许多方法都没有考虑到 AI 任务的计算特征,导致对于计算时延的估计并不准确。此外,在处理传输延迟时,大多将其简化为数据量与传输速率的比值,没有考虑多路径传输下的传输时延,这些都导致任务的响应时延增加。

2.4　预测计算时延

在计算时延预测领域,早期模型基于 FLOPS[38]进行计算时延估计。然而,它忽略了神经网络架构的复杂性,导致预测任务执行时间的误差较大。随后,研究者提出了更精细的分层时延预测方法[39],但这些方法仍简化了操作时延的计算,未能充分捕捉计算过程中的复杂操作。文献[40]和文献[41]分别通过端到端训练和元学习技术,提升了预测的准确性和效率。文献[42]进一步通过统一编码和硬件嵌入增强了模型的适应性。尽管如此,这些方法在处理大量任务和设备时,仍面临着数据效率和泛化能力的双重挑战。

尽管上述方法在计算时延预测方面取得了进展,但是它们未能解决开放网络环境下新设备上的任务计算时延预测的问题。当新设备到来,无法使用原有的模型来预测任务在新设备的计算时延,并且由于新设备的样本数量有限,难以基于这些样本训练新模型以解决时延预测问题。

2.5　总　结

总结来看,流量工程主要关注路径规划而不考虑节点选择,旨在通过优化路由策略提高网络性能;而任务卸载则重视节点选择,强调计算能力的适应性,却相对忽视路径规划对整体性能的影响。本文提出了一种紧耦合的算力网络控制系统,通过智能决策,实现对时延敏感型任务的高效调度,从而提供更优质的网络服务。

3　方　法

3.1　方法总述

传统的调度方法通常采用单一策略,导致计算资源和网络资源分配之间存在解耦合问题,无法满足时延敏感型任务的时延需求。因此,如何实现计算资源和网络资源的协同调度,以最小化响应时延,成为了一个关键挑战。

为解决这一挑战,本文构建了一个高效的算力

网络调度系统,该系统采用基于强化学习的联合优化策略,旨在协同优化计算资源(算力节点的选择)和网络资源(路径规划)的分配,提高任务执行效率并降低时延。系统实时获取网络状态和算力资源分布状态信息,使算力网络控制器能够做出智能决策,确保了在满足任务需求的同时,最小化任务响应时延,显著提升用户体验。

如图 1 所示,算力网络架构分为两个部分:控制平面和执行平面。

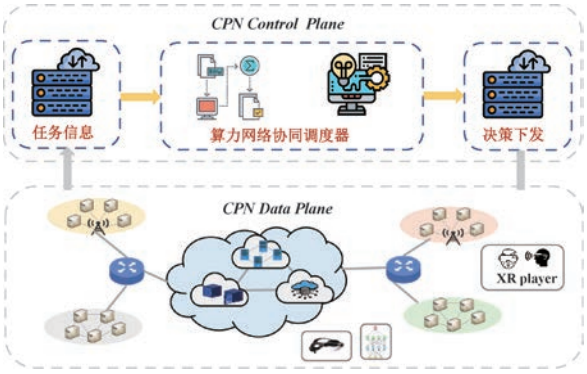


图 1 算力网络管理和调度框架图(RLMS-CPN)

算力网络控制平面部署了基于强化学习的算力网络协同调度器。控制平面接收执行平面上传的任务信息后,获取计算资源特征(算力时延估计)和网络资源特征(网络路径状态探测),输入强化学习模型,输出调度决策,并下发至执行平面,指导任务的传输与计算。

算力网络执行平面包含多个算力节点,负责算力任务的执行,包括任务数据的传输和在算力节点的执行过程。执行平面将任务信息上传至控制平面,获取调度策略,并根据控制平面的指令完成任务的传输与计算,确保任务高效执行。

在算力网络的实际应用中,AI 任务扮演着至关重要的角色。如图 2 所示,AI 任务基于多样化的神经网络架构,不同神经网络模型在隐藏层的层级数量、类型(如全连接层、卷积层、注意力机制层等)以及连接方式(如前馈结构、循环连接、残差跳跃等)上存在显著差异。并且,AI 任务在不同设备上运行效率差异显著,这与设备属性紧密相关。常见设备如 CPU、GPU 和 FPGA 等,它们各自有独特优势。CPU 通用性强,擅长逻辑控制和顺序处理,在 AI 数据预处理和模型训练流程控制方面表现良好。GPU 则凭借强大的并行计算能力,能快速处理大规模矩阵运算,在神经网络的训练和推理中发挥重要作用,加速模型运行。FPGA 能依据 AI 任务需求灵活配置硬件逻辑,在实时性要求高的场景,能迅速处理数据。

因此,算力网络需依据 AI 任务需求,以及不同算力节点的计算能力和当前的网络状态,为其分配传输路径与计算节点,提升 AI 任务的运行效率。

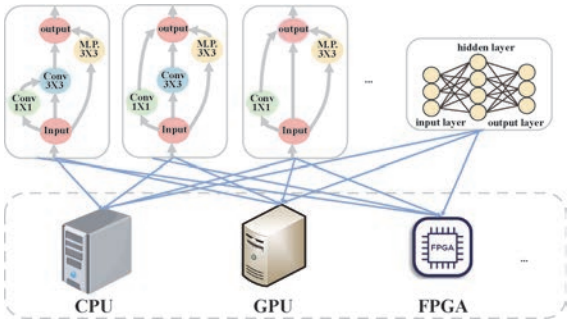


图 2 AI 任务示意图

综上,为实现上述算力网络协同调度,特别是在处理 AI 任务时,需要解决以下问题和挑战:

(1)计算时延估计:在任务调度前,需获取当前计算资源的状态信息。计算时延是指任务在算力节点上所需的计算时间。由于 AI 任务中神经网络的复杂性,无法使用简单的 FLOPS 方法预测任务的计算时延。本文利用神经网络来进行计算时延的估计。

然而,在开放算力网络环境中,算力网络动态变化,新设备会不断加入。首先,现有的计算时延估计模型可能无法适应新设备的特性;其次,由于缺乏新设备样本数量,不足以为新设备重新训练新的模型。因此,设计一种能够针对开放空间中新设备进行准确计算时延估计的模型是一个重要的挑战。

(2)如何实现协同调度,最小化任务的响应时延:在协同调度的目标是最小化任务的响应时延。节点的选择会影响任务的计算时延和传输时延;路径规划会影响任务的传输时延。本文利用计算资源特征和网络资源特征进行节点选择和路径规划调度。节点选择受计算资源和网络资源特征影响;路径规划受网络资源特征影响。并且节点选择会影响路径规划的路径对象。如何对这种复杂的关联性建模,以进行有效的特征融合,优化这两个任务,是一个重要的挑战。

(3)大规模网络环境下的调度策略可扩展性:在大规模网络环境中,调度策略的可扩展性是确保系统能够高效运行的关键。随着网络规模的扩大,资源调度决策的空间增加,导致调度的时间复杂度增加。调度系统必须具备良好的扩展性,以应对算力网络日益增加的复杂性和动态变化。

大规模网络中,计算节点的管理和实时网络状态变得更加复杂。调度策略必须能够处理这些复杂关系,确保任务的低时延需求。大规模网络需要处理海量的状态数据,如何高效收集、分析这些数据,以支持任务的高算力和低时延需求,是可扩展性的另一个难点。

针对上述问题和挑战,基于强化学习的算力网络调度的具体流程如图 3 所示,其中细化了算力网络控制平面的结构,具体调度流程主要包含以下步骤:

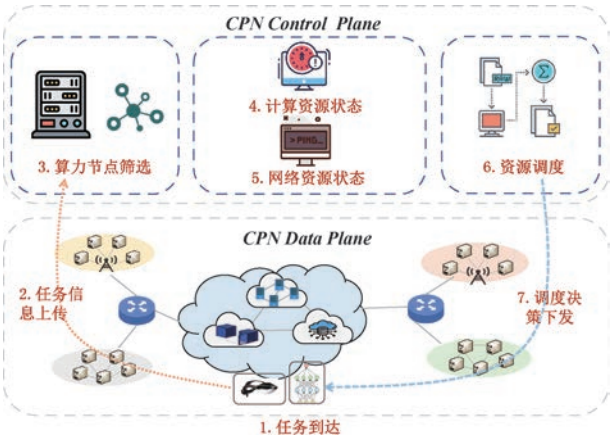


图 3　算力网络调度流程图

任务到达与信息上传:算力网络设有多个接入点,随时可能接收新的业务请求。当时延敏感型任务到达时,执行平面将任务相关信息上传至控制平面。

筛选算力候选节点:为了确保在大规模网络中实现良好的扩展性,调度器在接收任务信息后,会根据当前的计算资源和网络资源,对算力节点进行初步筛选,排除计算能力不足或网络传输成本过高的节点。(见 3.2 节)

计算时延预测:在开放空间下,提出两阶段微调策略(TSFT),以提高新设备计算时延的预测准确性。(见 3.3 节)

探测网络路径状态:通过发送探测数据包,获取当前路径的单向延时,捕获算力网络的状态信息。(见 3.4.2 节)

基于强化学习的算力网络资源调度:基于计算资源特征和网络资源特征,提出新的 GNN 架构为调度问题进行高效的特征融合与提取,最后利用强化学习模型进行算力网络的资源调度,确定节点的选择以及路径规划。(见 3.4 节)

调度指令下发:控制平面将最终调度指令传递给执行平面,完成时延敏感型任务的传输与计算。

3.2　算力候选节点筛选

在大型算力网络中,调度策略的可扩展性面临挑战,因为算力节点选择的空间非常庞大,且路径规划方式更加复杂,这会导致调度决策的时间开销显著增加。对于时延敏感型任务而言,降低调度方案的计算时间,确保及时响应和优化任务执行至关重要。

为了有效解决这一可扩展性问题,调度器在接收到任务信息时,包含任务到达的接入点位置等关键数据,首先进行算力节点的筛选,以缩小调度策略的搜索空间,减少调度方案的时间开销。筛选方案需要综合考虑计算资源和网络资源,旨在减小调度空间的同时,保证调度策略的高效性。

(1)计算资源:预测任务在不同类型算力平台上的计算时延,以选择合适的算力执行平台。例如,对于新任务预测其在各种算力平台上的计算时延,如果结果显示该任务更适合在 GPU 设备上执行,首先筛选出所有的 GPU 节点,并在此基础上进一步考虑网络资源。

(2)传输资源:根据任务到满足计算条件的算力节点的最短路径的跳数来选择算力网络候选节点。

(3)生成算力邻居拓扑图:算力网络调度器基于网络的全局视图信息,综合考虑计算资源和网络资源,形成算力邻居拓扑图(如图 4 所示)。该拓扑图包含任务的接入点到所有候选算力节点之间的网络拓扑的信息。

图 4　算力节点筛选示意图

通过筛选高效的算力节点,显著降低了调度决策空间,从而降低任务调度的时间复杂度。形成的算力邻居拓扑图将被传递给算力网络控制器,后者将在此基础上进一步进行任务调度。在调度之前,需要获取计算资源信息,估计任务在不同算力节点上的计算时延。

3.3　面向开放空间下的计算时延估计模型

为了获取当前计算资源信息,需要对任务在不同设备上的计算时延进行估计。计算时延,即任务在算力节点上所需的计算时间,是响应时延的关键组成部分。鉴于 AI 任务中神经网络的复杂性,传统的 FLOPS 方法已不足以准确预测任务的计算时延。因此,本文采用基于神经网络的方法来预测任务在不同算力设备上的计算时延。本文利用已有的

设备样本信息训练一个基础模型，以估计现有设备的计算时延，从而辅助优化调度策略。然而，在开放环境下，新的算力设备不断加入，基础模型难以精确预测新设备上的计算时延。为了克服这一挑战，本文设计了一种两阶段微调策略（TSFT），旨在提高新设备计算时延的预测准确性。

基本时延预测模型。如图 5 所示，对于新型 AI 应用，获取到任务的神经网络结构，包括它们的基本组成和连接方式。基于这些信息，将任务的神经网络结构编码为两个矩阵（“邻接－操作矩阵”）。

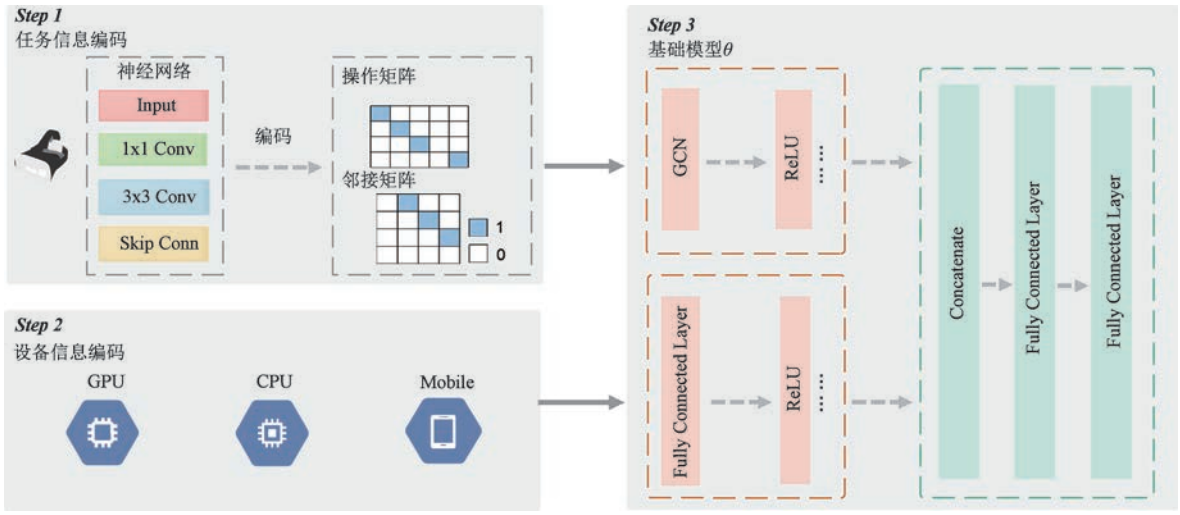


图 5 基础模型流程图

任务信息编码。本文将神经网络架构抽象为两个核心组成部分：邻接矩阵和操作矩阵。邻接矩阵用于描述网络层之间的连接关系。具体来说，第 0 行第 1 列为 1，表示网络层 0 连接到网络层 1；同理，第 1 行第 2 列为 1，表示网络层 1 连接到网络层 2，以此类推，编码整个网络的结构。假设神经网络存在 5 种操作方式：无操作（Null）、1x1 卷积（Conv1x1）、3x3 卷积（Conv3x3）、3x3 平均池化（AvgPool3x3）、跳跃连接（Skip-connection）。对于操作矩阵，每一行表示对应网络层所执行的操作类型。例如：第 1 行第 1 列的位置代表 1x1 卷积操作。在完成邻接矩阵和操作矩阵的编码后，将这些信息整合，将其作为任务的状态信息 X。

训练目标。假设共有 $p(\tau)$ 个设备，对不同硬件设备进行编码，V_h^τ 表示设备的特征，我们将设备信息和任务信息同时输入到基础模型 $f(\theta)$ 中进行训练。训练目标如下所示：

$$\min_{\theta} \sum_{\tau \sim p(\tau)} \mathcal{L} L(f(X, V_h^\tau; \theta), Y^\tau) \tag{1}$$

其中，$f(X^\tau, V_h^\tau; \theta)$ 表示基础模型预测的计算时延，Y^τ 表示样本的真实时延。

面向开放空间的两阶段微调策略（TSFT）。在开放的算力网络环境中，随着新设备的动态加入，原有设备模型无法准确预测任务在新设备上的计算时延。因此，如何在新设备加入算力网络时进行准确的计算时延预测，成为一项重要挑战。

为了应对这一挑战，本文提出了一种新的解决方案。在基础时延预测模型之上，提出两阶段微调策略（TSFT），训练流程如图 6 所示。

（1）第一阶段微调：在这一阶段，使用少量带标

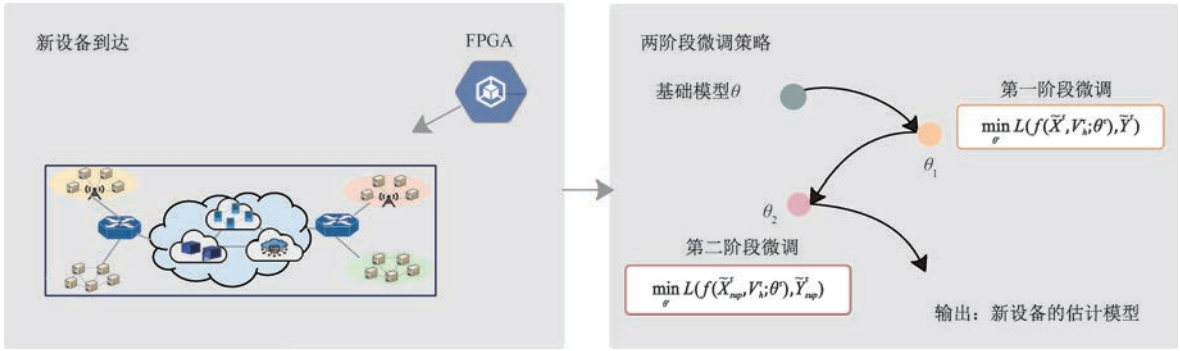


图 6 两阶段微调策略

签的新设备样本进行模型训练,以优化新设备的计算时延预测模型。训练目标如下所示:

$$\min_{\theta^{\tau}} \mathcal{L}(f(\widetilde{X}^{\tau},V_h^{\tau};\theta^{\tau}),\widetilde{Y}^{\tau}) \tag{2}$$

其中,$f(\widetilde{X}^{\tau},V_h^{\tau};\theta^{\tau})$ 表示模型估计的新设备的计算时延,$\widetilde{X}^{\tau}$ 表示少量新设备的样本,V_h^{τ} 表示新设备的特征表示,θ^{τ} 表示模型参数,$\widetilde{Y}^{\tau}$ 表示样本的真实时延。

(2)第二阶段微调:在模型训练完成后,基于上述模型,对于任务的样本 $\widetilde{X}^{\tau}_{\text{sup}}$,输入到上述模型得到伪标签[43] $\widetilde{Y}^{\tau}_{\text{sup}}=f(\widetilde{X}^{\tau}_{\text{sup}},V_h^{\tau};\boldsymbol{\theta}^{\tau})$,利用这些样本再次对模型进行训练。训练目标如下所示:

$$\min_{\theta^{\tau}} \text{L}(f(\widetilde{X}^{\tau}_{\text{sup}},V_h^{\tau};\theta^{\tau}),\widetilde{Y}^{\tau}_{\text{sup}}) \tag{3}$$

通过上述训练方法可以构建出有效的新设备计算时延的估计模型。接下来详细阐述本文的算力网络资源调度策略。

3.4　基于强化学习的算力网络资源调度

在算力网络中,资源调度对于优化任务执行效率至关重要,尤其是对于时延敏感型任务,快速且准确的调度策略是降低响应时延的关键。然而,传统调度方法往往在计算资源和网络资源分配之间解耦,难以实现整体优化,无法满足任务的响应时延需求。

为了最小化任务响应时延,协同调度框架通过特征融合解决节点选择和路径规划问题。节点选择依赖于计算和网络资源特征,而路径规划则主要受网络资源特征影响。节点选择的结果又会影响路径规划的路径对象。有效地建模这些复杂的关联性,以优化任务的计算和传输时延,是一个重大挑战。

为了应对这一挑战,本文提出了一种新的紧耦合算力网络资源调度策略。该方案通过构建 SyncGNN 模型,将计算特征和网络特征深度融合,并提取这些融合特征作为调度模型的输入。总体调度决策流程如图 7 所示,分为以下两个步骤:

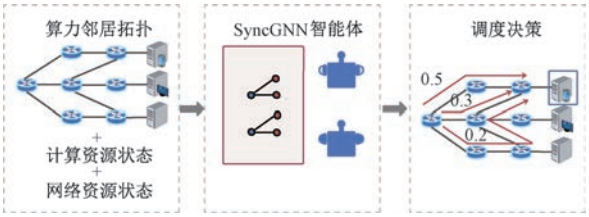


图 7　整体框架

(1)构建 SyncGNN 融合算力特征和网络特征

本文设计了一种新的 GNN 架构(SyncGNN),通过将算力节点和传输路径构建为 GNN 中的“节点”,表示当前算力网络的计算资源和网络资源信息。利用 SyncGNN 的信息交换技术,进行算力节点与路径节点之间的信息传递,提取算力节点选择和路径规划任务的关键特征。

(2)基于强化学习的算力网络资源调度

在 SyncGNN 完成特征融合后,本文使用强化学习进行资源调度,智能体通过与环境交互学习调度策略,以最小化时延敏感型任务的响应时延。为了应对节点选择和路径规划问题,本文分别设计了节点选择智能体和路径规划智能体,分别负责算力节点的选择和路径规划,协同提高决策性能。

3.4.1　构建 SyncGNN

为了同时考虑算力节点选择和路径规划问题,本文采用 GNN 来融合计算资源和网络资源的特征。GNN 是专为处理图结构数据的神经网络[44],在多个领域得到广泛应用,包括网络规划[45]、社交网络[46-47]和流量预测[48]。GNN 通过信息传递机制学习节点嵌入特征[49-50]。

尽管 GNN 具有许多优势,在算力网络资源调度中,本文关注的核心是节点选择和路径规划问题。因此,本文将重点放在与调度相关的实体:算力节点和路径上,并将它们抽象为 GNN 中的节点。基于此,本文构建了一种新的 GNN 架构——SyncGNN,如图 8 所示。

当新任务到达算力网络接入点 S1,首先为其筛选算力候选节点,形成算力邻居拓扑图。对于每个算力候选节点,创建一个“算力节点”(例如 T1、T2、T3);对于每条相关路径,创建一个“路径节点”(例如,对于从算力网络接入点 S1 到算力节点 T1 的三条路径:P^{125}:S1－＞S2－＞S5、P^{135}:S1－＞S3－＞S5、P^{14735}:S1－＞S4－＞S7－＞S3－＞S5,分别为这三条路径创建路径节点:P^{125}、P^{135}、P^{14735})。

在 SyncGNN 中,只有当路径连接算力网络接入节点和算力节点时,算力节点和路径节点才会相连。蓝色的节点代表算力节点,使用计算时延进行初始化;红色的节点代表路径节点,使用当前路径的信息进行初始化。

在进行特征融合之前,需要获取当前路径的信息,以初始化路径节点。

3.4.2　算力网络状态信息提取

为了获取网络中各路径的状态信息,本文采用主动探测的方式,沿不同路径发送探测包。主动探测的频率及大小都会影响测试结果,越高的探测频率和越多探测次数会带来更准确的探测结果,但是

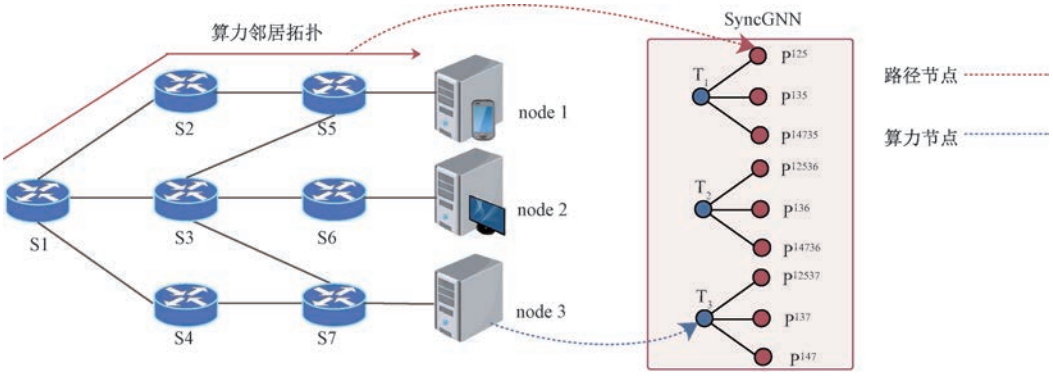


图 8 构建 SyncGNN

会给网络带来更多负担。综合考虑探测准确性和额外开销，本文采用连续背靠背发送三次探测包进行探测，对三次探测结果取平均值避免单次探测结果偶然性的影响。本文以探测包的平均单向时延作为当前路径的状态，以此捕捉网络状态信息。单向时延能够直观地反映路径上的网络延迟情况，从而为后续的路径选择和任务调度提供依据。

同时针对后续的数据传输，为了实现多路径的并行传输，本文在 NS3[51] 中采用了 Flowlet[52] 技术，该技术将数据流拆分成多个较小的“流片”进行调度。Flowlet 技术支持在多路径上并行传输，有效减少网络拥塞和时延。

具体实现如图 9 所示，任务的源节点为 S1，目标节点为 S5，从 S1 到 S5 有三条路径：P1(S1→S2→S5)、P2(S1→S3→S5)和 P3(S1→S4→S7→S3→S5)。按照比例将数据包拆分成多个 Flowlet 并沿不同路径传输。

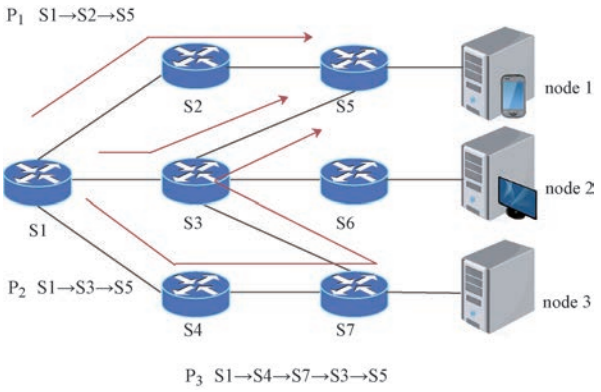


图 9 多路径传输

如表 1 所示，在节点 1 的路由表中，对于从源节点 S1 到目标地址 S5 的数据流，假设 50% 的数据沿路径 P1 传输，30% 的数据沿路径 P2 进行传输，20% 的数据沿路径 P3 进行传输。对于 50% 的数据包，设置 flag=0，沿路径 P1 传输；对于 30% 的数据包，设置 flag=1，沿路径 P2 传输；对于 20% 的数据包，设置 flag=2，沿路径 P3 传输。采用 Flowlet 技术能够根据调度决策的路径分流比，灵活调整不同路径上的传输流量比例，实现任务传输的高效和灵活性。

表 1 节点 1 的路由表

源地址	目标地址	flag	下一跳
S1	S5	0(P1)	S2
S1	S5	1(P2)	S3
S1	S5	2(P3)	S4

3.4.3 SyncGNN 特征融合

在获取算力网络路径状态信息后，需要进一步进行特征融合，以便在进行算力节点选择的同时考虑任务的计算时延和传输时延。

在 SyncGNN 中，拓扑结构可定义为 $G=(V,E)$，其中 V 代表节点集。SyncGNN 包含两类节点：算力节点 v_i^c 和路径节点 v_i^p，h_i^c 表示算力节点 v_i^c 的特征，使用计算时延初始化；h_i^p 表示路径节点 v_i^p 的特征，使用单向时延初始化。E 是连接节点之间的边集，在 SyncGNN 中，当且仅当路径连接算力网络接入节点和算力节点时，算力节点和路径节点才会相连。对于每个算力节点 v_i^c 与其相连的路径节点表示为 $N(v_i^c)$。

SyncGNN 采用消息传递函数进行特征融合。算力节点和相连的路径节点之间只需进行一次信息交换，即可将网络信息传递给算力节点。SyncGNN 的信息交换过程包括两个阶段，聚合与更新，如图 10 所示。

在聚合阶段：对于每个算力节点，首先获取相邻路径节点的信息 m_i：

$$m_i=\sum_{v_j\in N(v_i^c)} M(before\_h_i^c,h_j^p) \tag{4}$$

其中，$before\_h_i^c$ 表示更新之前的算力节点特征，M

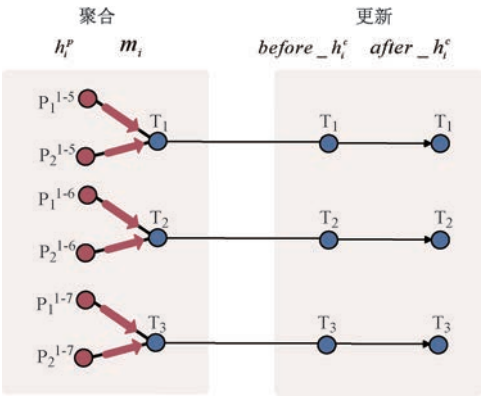


图 10　信息交换

表示聚合函数。

在更新阶段:获得邻居路径节点信息 m_i 之后,更新算力节点的特征,如下所示:

$$after\_h_i^c = U(before\_h_i^c, m_i) \tag{5}$$

其中,$after\_h_i^c$ 表示更新之后的算力节点特征,U 表示更新函数。

通过上述信息交换方式,确保每个算力节点的特征不仅包含计算资源的信息,还包含网络资源(任务接入点到算力节点的路径)的信息。在利用算力节点的特征进行算力节点的选择时,将同时考虑计算资源和网络资源,从而选择更合适的算力节点。

综上所述,利用 SyncGNN 进行特征融合,提取算力节点特征 h_i^c 与路径节点特征 h_i^p,作为当前网络的状态信息,输入到后续的调度模型中。

3.4.4　基于强化学习的算力网络的资源调度

本文采用强化学习来建模算力网络资源调度问题,具体流程如图 11 所示。对于新到达的时延敏感型任务,智能体根据当前状态 $s_t = \{h_t^c, h_t^p\}$,输出一个可行的动作 $a_t = \{a_t^c, a_t^p\}$。具体来说,节点选择智能体(智能体 1:π^c)输出算力节点的选择决策 $a_t^c = \pi^c(h_t^c; \theta^c)$;路径规划智能体(智能体 2:$\pi^p$)输出路径规划方式 $a_t^p = \pi^p(h_t^p; \theta^p)$。智能体将调度决策指令下发给环境执行任务的调度,环境将奖励反馈给智能体。具体概念如下:

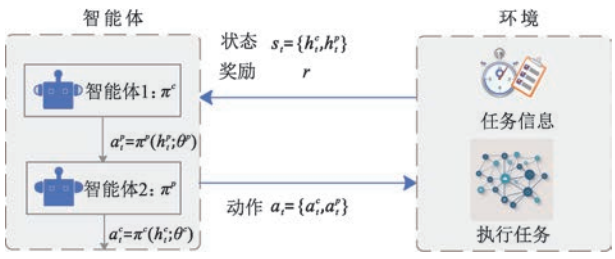


图 11　强化学习对算力网络资源调度问题建模

(1)状态空间:在调度系统中,当前状态由任务和网络共同决定。本文将计算资源信息和网络资源信息传递给 SyncGNN 进行特征融合,输出节点和路径特征。系统在 t 时刻的状态定义为

$$s_t = \{h_t^c, h_t^p\} \tag{6}$$

其中,h_t^c 表示在时间 t 算力节点的特征;h_t^p 表示在时间 t 路径节点特征。

(2)动作空间:智能体根据系统当前的状态信息,在时刻 t 选择执行的动作,包括算力节点的选择和路径规划。动作可以表示为

$$a_t = \{a_t^c, a_t^p\} \tag{7}$$

值得注意的是,调度策略的动作空间包括离散动作和连续动作,节点选择为离散动作 $a_t^c = \pi^c(h_t^c; \theta^c)$,假设有 n 个算力候选节点,节点选择智能体需要从中选择一个算力节点作为计算节点,算力节点选择的动作空间大小为 n;路径规划方式为连续动作 $a_t^p = \pi^p(h_t^p; \theta^p)$,假设每个节点对有 k 条路径,路径规划需要规定 k 条路径的分流比,并且 k 条路径的分流比之和为 1。

(3)奖励:反映了智能体在当前状态下的动作效果。奖励公式如下所示:

$$r_t = -t_{resp} \tag{8}$$

其中,t_{resp} 为任务的响应时延。r_t 反映了任务调度决策的质量,当响应时延越低,奖励值越大,表示当前调度决策质量越好。

训练阶段

SyncGNN 和智能体共同构成了调度模型。本文采用 Actor-Critic[53] 算法端到端训练模型,目标是最小化时延敏感型任务的响应时延。Actor-Critic 算法结合了策略梯度方法和值函数估计的强化学习算法,通过 Actor 网络直接学习策略以选择最优动作,同时 Critic 网络评估当前策略下的状态值,两者相互协作以优化整体性能。具体的调度模型训练过程如算法 1 所示。

输入任务信息和当前网络信息,输出得到训练好的节点选择智能体 π^c 和路径规划智能体 π^p,能够有效地根据任务需求和网络条件做出高效的资源调度决策。

Parameter initialization

首先随机初始化节点选择智能体 π^c 的参数 θ^c,路径规划智能体 π^p 的参数 θ^p 和价值网络 Q 的参数 φ。

State generation

在 t 时刻,当时延敏感型任务 $task_t$ 到来,我们首先经过算力节点筛选,计算时延预测,网络路径状

态提取和 SyncGNN 特征融合，得到当前状态 $s_t=\{h_t^c,h_t^p\}$。

Action generation

两个智能体根据当前网络状态 s_t 生成调度决策 $a_t=\{a_t^c,a_t^p\}$。

Execute task

下发调度决策指令给执行平面。执行平面完成任务的传输和计算，得到奖励 r_t 与下一个时刻状态 s_{t+1}。Critic 网络对状态进行评分：

$$Q_t=Q(s_t;\varphi_{\text{now}}) \text{ and } Q_{t+1}=Q(s_{t+1};\varphi_{\text{now}}) \tag{9}$$

其中，Q_t 表示 Critic 网络对 s_t 的评分，Q_{t+1} 表示 Critic 网络对 s_{t+1} 的评分。

Critic 网络通过时序差分目标（Temporal Difference Target，简称 TD 目标）进行参数更新，其计算公式如下所示：

$$y_t=r_t+\gamma\cdot Q_{t+1} \tag{10}$$

其中，r_t 为当前奖励，γ 表示衰减因子，Q_{t+1} 表示 Critic 网络对 s_{t+1} 的评分。TD 目标 y_t 表示对当前状态 s_t 的评分，并且由于 y_t 部分基于实际观测到的奖励 r_t，认为 y_t 比 Q_t 更加接近事实真相。将 y_t 和 Q_t 之间的差值定义为 δ_t，称为 TD 误差，计算公式如下所示：

$$\delta_t=Q_t-y_t \tag{11}$$

Parameter updating

本文希望 Q_t 尽可能接近 y_t，所以使用梯度下降更新价值网络 $Q(\varphi)$：

$$\varphi_{\text{new}}\leftarrow\varphi_{\text{now}}-\alpha\cdot\delta_t\cdot\nabla_{\varphi}Q(s_t;\varphi_{\text{now}}) \tag{12}$$

其中，当前价值网络参数为 φ_{now}，执行梯度下降更新网络参数为 φ_{new}，α 表示价值网络的学习率。

智能体学习的目标在于学习一个好的策略，对于所有状态，智能体都能给出好的策略，得到高的奖励值，智能体的训练过程可以表述为下面的优化问题：

$$\max_{\theta}\{J(\theta)\triangleq\mathbb{E}_{r(s,a)\sim\pi_\theta}[r(s_t,a_t)]\} \tag{13}$$

求解这个最大化问题最简单的算法就是梯度上升。设当前节点选择智能体 $\pi^c(\theta^c_{\text{now}})$ 的参数为 θ^c_{now}。做梯度上升更新参数，得到新的参数 θ^c_{new}：

$$\theta^c_{\text{new}}\leftarrow\theta^c_{\text{now}}+\beta_c\cdot\delta_t\cdot\nabla_\theta\ln\pi^c(a_t\mid s_t;\theta^c_{\text{now}}) \tag{14}$$

其中，β_c 表示节点选择策略网络的学习率。

路径规划智能体更新方式同理：

$$\theta^p_{\text{new}}\leftarrow\theta^p_{\text{now}}+\beta_p\cdot\delta_t\cdot\nabla_\theta\ln\pi^p(a_t\mid s_t;\theta^p_{\text{now}}) \tag{15}$$

其中，β_p 表示路径规划策略网络的学习率。

当训练完成后，算法输出两个训练好的智能体，其中节点选择智能体 π^c 能够选择合适的算力节点；路径规划智能体 π^p 能够合理地规划传输方式，它们能够对不同的状态做出好的调度决策。

算法 1

输入：任务信息和当前网络信息

输出：节点选择智能体 π^c，路径规划智能体 π^p

1 Parameter initialization

2 初始化节点选择智能体参数 θ^c；初始化路径规划智能体参数 θ^p；初始化 Critic 网络参数 φ

3 FOR $episode=1,2,\cdots\cdots,N$ DO

4 FOR $epoch=1,2,\cdots,T$ DO

5 State generation

6 根据当前任务 $task_t$ 信息和当前网络信息利用 SyncGNN 生成当前状态特征：

$$s_t=\{h_t^c,h_t^p\}$$

7 Action generation

8 节点选择智能体进行节点选择

$$a_t^c=\pi^c(h_t^c;\theta^c)$$

9 路径规划智能体进行路径规划

$$a_t^p=\pi^p(h_t^p;\theta^p)$$

10 Execute task

11 得到奖励值 r_t 和下一个状态 s_{t+1}

12 Critic 网络对 s_t 和 s_{t+1} 进行评分

$$Q_t=Q(s_t;\varphi_{\text{now}}) \text{ and } Q_{t+1}=Q(s_{t+1};\varphi_{\text{now}})$$

14 计算 TD 目标和 TD 误差：

$$y_t=r_t+\gamma\cdot Q_{t+1} \text{ and } \delta_t=Q_t-y_t$$

15 Parameter updating

16 更新 Critic 网络：

$$\varphi_{\text{new}}\leftarrow\varphi_{\text{now}}-\alpha\cdot\delta_t\cdot\nabla_{\varphi}Q(s_t;\varphi_{\text{now}})$$

17 更新节点选择智能体：

$$\theta^c_{\text{new}}\leftarrow\theta^c_{\text{now}}+\beta_c\cdot\delta_t\cdot\nabla_\theta\ln\pi^c(a_t\mid s_t;\theta^c_{\text{now}})$$

18 更新路径规划智能体：

$$\theta^p_{\text{new}}\leftarrow\theta^p_{\text{now}}+\beta_p\cdot\delta_t\cdot\nabla_\theta\ln\pi^p(a_t\mid s_t;\theta^p_{\text{now}})$$

19 ENDFOR

20 END FOR

执行阶段

训练完成后，运用训练好的策略网络 $\pi=\{\pi^c,\pi^p\}$ 进行实时调度决策。整个算力网络调度流程如图 12 所示：

（1）当时延敏感型任务到来时，其信息首先会被传送至算力网络的控制层（步骤 1-2）。

（2）算力网络控制平面的调度器根据全网的计算资源和网络资源，筛选出合适的计算节点，排除掉

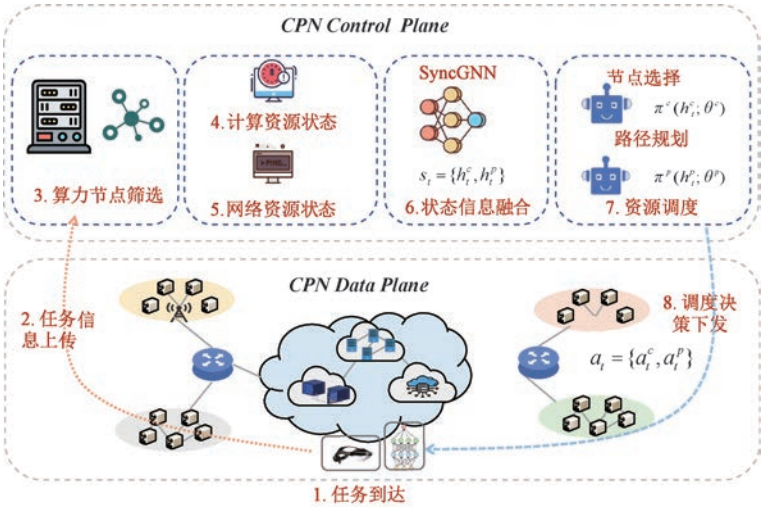


图 12　算力网络调度流程图

那些计算能力不足或网络传输成本太高的节点。调度器将筛选后的候选节点及任务信息发送给网络控制器。控制器预测节点的计算时延并且探测路径的传输时延,这些信息输入到 SyncGNN 进行特征融合,生成当前状态 $s_t=\{h_t^c,h_t^p\}$。智能体 $\pi=\{\pi^c,\pi^p\}$ 根据当前状态,输出任务的调度决策 $a_t=\{a_t^c,a_t^p\}$,即算力节点的选择和路径规划(步骤 3-7)。

(3) 最后,网络控制层将调度指令下发给算力网络执行平面,执行任务调度。任务沿着指定的路径分流比被调度到指定的算力节点进行计算。任务的计算时延为任务在算力节点的计算时间,任务传输时延为沿不同路径的最大传输时延,两者之和为任务总的响应时延(步骤 8)。

4　实　验

4.1　实验设置

在本节中,通过一系列基于真实世界网络拓扑结构的模拟实验来评估方案的性能,并与其他基线方案进行比较,以展示其有效性。所有实验均在 NS3 [51] 仿真环境中进行。本文从以下几个关键维度来评估算法的性能:

响应时延:评估方法在不同网络拓扑上的响应时延,并与基线方案进行比较,以验证协同调度的有效性(见 4.2 节)。

SyncGNN 模型的有效性:在不同的网络拓扑中比较 SyncGNN 模型和 MLP 对响应时延的影响(见 4.3.1 节)。

计算时延估计的准确性:评估提出的两阶段微调算法(TSFT)在开放空间下对新设备计算时延估计的准确性(见 4.3.2 节)。

参数敏感性:评估算力候选节点数量对响应时延的影响(见 4.4.1 节),以及强化学习超参数设置(见 4.4.2 节)。

大规模算力网络的可扩展性:评估方法在大规模算力网络的表现(见 4.5 节)。

调度方案的计算时间开销:评估方法的计算时间开销(见 4.6 节)。

任务类型对调度性能的影响:评估方法在不同任务类型下的响应时延(见 4.7 节)。

4.1.1　数据集

实验基于四个真实世界的网络拓扑结构,包括 Abilene 网络[54],以及 RocketFuel[55]收集的三个服务提供商网络(即 EBONE、Sprintlink 和 Tiscali)。四种拓扑的节点数和链路数量详见表 2。

表 2　网络拓扑信息

网络拓扑	节点数量	链路数量
Abilene	12	30
EBONE(Europe)	23	76
Sprintlink(USA)	44	166
Tiscali	49	172

对于 AI 任务,通过 NAS-Bench－201 数据集为每个任务附加神经网络信息。本文选择了 5 个算力候选节点,3 条候选路径。在 4.5 节中,讨论了不同数量的算力候选节点对任务调度决策的影响。

为了考量分布式算力节点的配置和计算能力差异性对调度系统的影响,本文对于算力节点的硬件配置进行分类。主要包括 CPU、GPU 和其他设备,GPU 涵盖多种高性能显卡,如 NVIDIA GeForce GTX 1080 Ti、NVIDIA Titan X、NVIDIA Titan

RTX、NVIDIA GeForce RTX 2080 Ti 和 NVIDIA Titan XP 等;CPU 包含多种 CPU 服务器,如 Intel Xeon Silver 4114、Intel Xeon Silver 4210R、Intel Xeon Gold 6240、Intel Xeon Gold 6226 和 Intel Core i7-7820X 等。其他设备包括 FPGA、Raspberry Pi 4 和 Eyeriss 等。

在实验环节,针对每个网络拓扑结构,本文采用随机设置的方式,为每个节点分配不同计算能力的硬件配置,以此模拟真实世界中分布式算力节点的多样性。同时,随机设置了 2～4 个接入点作为任务到达算力网络的入口节点,负责接收任务。

以图 13 所示的 Abilene 网络的拓扑结构与算力设备为例,该网络包含 12 个节点与 30 条链路,其中 S8 和 S9 作为算力网络接入点,主要负责算力任务到达时,将任务信息上传给算力网络控制平面,其余 10 个节点分别配置了 5 个 CPU 与 5 个 GPU 设备,且各设备具备不同的计算能力。在 CPU 配置上,Intel Xeon Silver 4114、4210R 适合轻量级任务;Intel Xeon Gold 6240 和 6226 适用于高性能计算;桌面级的 Intel Core i7-7820X 则适合复杂计算任务。GPU 方面,NVIDIA GeForce GTX 1080 Ti 和 Titan X 显存较高(11GB/12GB),在深度学习领域表现出色;Titan RTX 和 RTX 2080 Ti 计算能力强,更适合复杂模型训练;Titan XP 则在显存和性能间取得平衡,适用于多种计算任务。通过这种对不同性能 CPU 和 GPU 设备的合理分配,算力网络的调度算法能够高效完成 AI 任务的传输与计算。

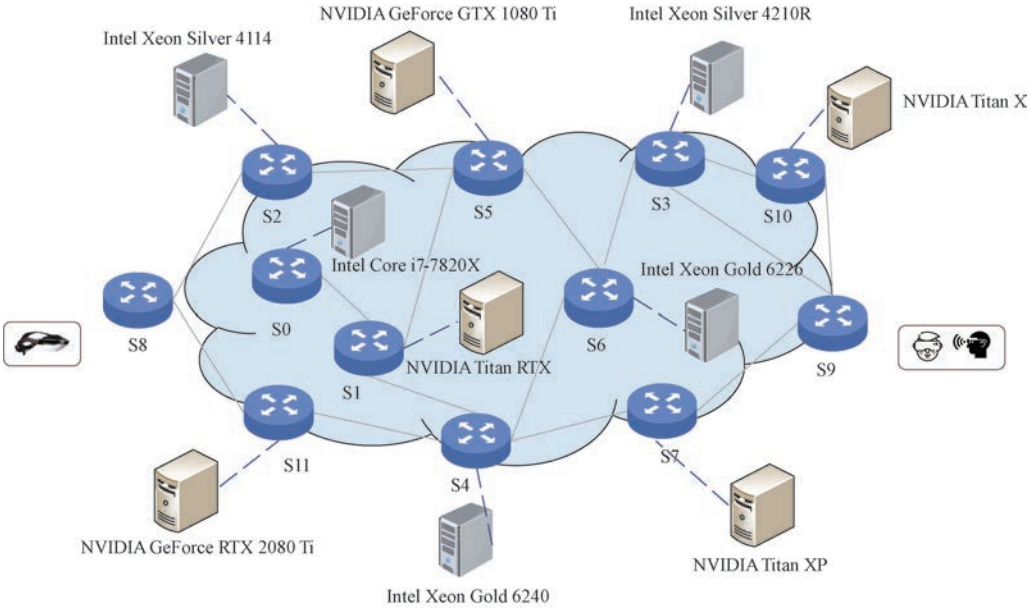


图 13 Abilene 算力网络结构

4.1.2 指 标

响应时延:该指标由计算时延和传输时延组成。RLMS-CPN 旨在通过协同调度最小化响应时延,响应时延越小,说明调度方案的效果越好。

均方误差(MSE):该指标用于衡量计算时延预测模型的准确性。MSE 值越低,表明预测值与真实计算时延之间的误差越小,模型的预测性能越好。

调度方案的计算时间开销:该指标指调度器生成决策方案所需的时间。计算时间越短,说明调度算法的实时性越强,越能满足算力网络对快速调度的要求。

4.1.3 基 线

据我们所知,没有针对 AI 任务同时进行计算时延和传输时延协同优化的研究内容,为了充分讨论,本文实现了如下基线方法进行比较。

(1)Base1:选择计算时延最小的算力节点[56-57],以确保任务能够尽快完成。其核心在于通过选择计算能力最强的节点来优化任务的计算时间。在数据传输方面,采用等价多路径(Equal-Cost Multi-Path,简称 ECMP)[15]策略,将数据均匀分配到三条最短路径上进行传输。

(2)Base2:选择最近的算力节点[32],以减少数据传输的物理距离,从而降低传输时延。该方法的核心是通过地理距离的优化来减少传输延迟。同样采用 ECMP[15]策略将数据在三条最短路径上均分传输。

(3)Base3:选择计算时延最小的算力节点[56-57],

同时在路径规划方面采用线性规划方法[13],目标是最小化最大链路利用率。该方法的核心在于通过计算优化和流量分配优化相结合的方式,在确保任务快速执行的同时,通过线性规划优化路径上的流量分配,避免链路过载。

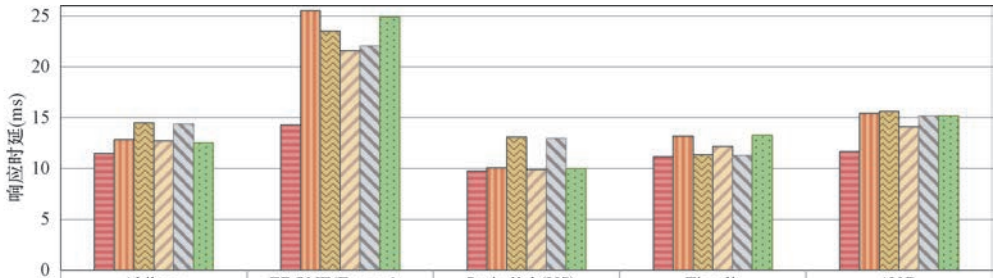
(4)Base4:选择最近的算力节点[32],同时在路径规划方面采用线性规划方法[13],目标是最小化最大链路利用率。该方法的核心在于通过地理距离优化和流量分配优化相结合的方式,在减少传输延迟的同时,通过线性规划优化路径上的流量分配,避免链路过载。

(5)GNN+DQN[8]一种面向算力感知网络的多维度资源分配机制,结合 GNN 和深度强化学习。

4.2　性能评估

4.2.1　平均响应时延

为了评估 RLMS-CPN 是否能够有效降低时延敏感型任务的响应时延并实现协同调度,在四个网络拓扑中测试了 100 个任务,所得每个网络拓扑下的任务平均响应时延结果如图 14 所示。



	Abilene	EBONE(Europe)	Sprintlink(US)	Tiscali	AVG
RLMS-CPN	11.4639916	14.30358029	9.731726004	11.16046709	11.66494125
Base1	12.83039904	25.54598192	10.07308204	13.20928186	15.41468621
Base2	14.47670531	23.50399355	13.10420739	11.3663208	15.61280676
Base3	12.71370209	21.59293371	9.862329899	12.19326709	14.0905582
Base4	14.40836354	22.05802815	12.96948137	11.28436237	15.18005886
GNN+DQN	12.52468211	24.89234234	9.980144124	13.27036736	15.16688398

图 14　四个网络拓扑的任务的平均响应时延

表 3 展示了 RLMS-CPN 在各个网络拓扑中相对于各个基线的提升水平。综合分析,RLMS-CPN 在所有测试的网络拓扑中均表现出显著优势。传统基线方法由于采用解耦合的调度方式,在调度时延敏感型任务时,显示出明显的不足,无法有效降低响应时延。GNN+DQN 方法的奖励函数在网络资源反馈方面主要涉及带宽,但带宽并不能直接优化传输时延,因此其效果相对较差。RLMS-CPN 通过实现协同调度,能够适应动态环境并优化任务调度,显著提高网络资源调度效率。

表 3　四个网络拓扑中相对于各个基线的提升

网络拓扑	Base1	Base2	Base3	Base4	GNN+DQN
Abilene	10.65%	20.81%	9.83%	20.44%	8.47%
EBONE	44.01%	39.14%	33.76%	35.16%	42.54%
Sprintlink	3.39%	25.74%	1.32%	24.96%	2.49%
Tiscali	15.51%	1.81%	8.47%	1.10%	15.89%
Avg	24.33%	25.29%	17.21%	23.16%	23.09%

4.2.2　每个网络拓扑下每个任务的响应时延

本文选取了 4 个不同的网络拓扑,分别展示了 100 个任务下的响应时延情况。如图 15-18 所示,

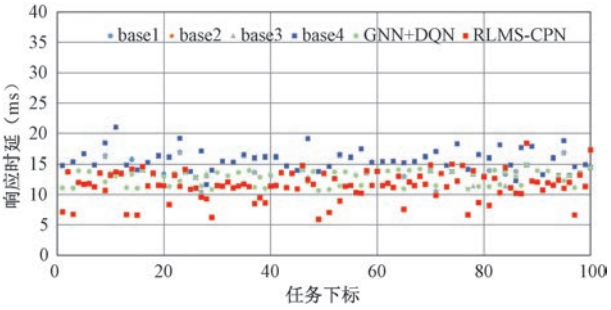


图 15　Abilene

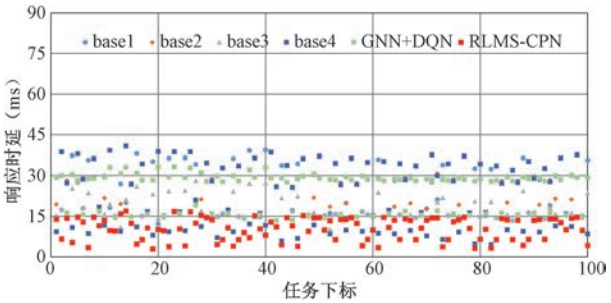


图 16　EBONE

纵坐标表示响应时延,横坐标为任务的下标,具体数值代表每个任务的实际响应时延。

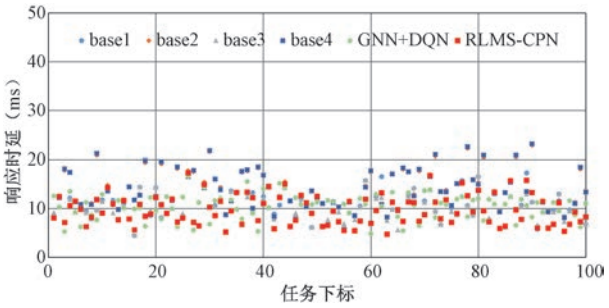


图 17 Sprintlink

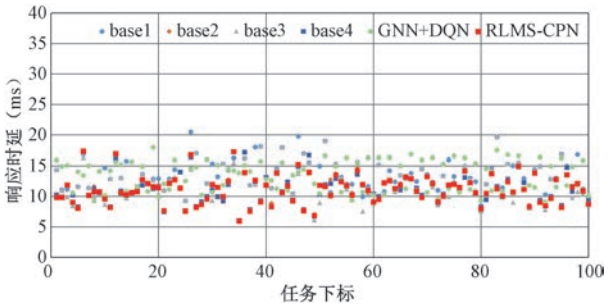


图 18 Tiscali

实验结果表明，在大多数任务下，RLMS-CPN 的响应时延明显优于对比方法。总体来看，在 100 个任务中，RLMS-CPN 的响应时延几乎始终低于其他方法，证明了 RLMS-CPN 能够实现计算资源和网络资源的协同优化，有效降低时延敏感型任务的响应时延。

4.3 消融实验

4.3.1 SyncGNN 特征融合有效性

为了验证 SyncGNN 在特征融合方面的有效性，在四个网络拓扑中分别训练了两种特征融合方式。在不使用 SyncGNN 的模型中，利用 MLP 进行任务的特征提取，除此之外，其他条件均保持一致。本文使用 100 个任务测试其响应时延，并记录了四个网络拓扑中任务的平均响应时延结果，详见表 4。

表 4 任务的平均响应时延

网络拓扑	SyncGNN	无 SyncGNN	提升百分比
Abilene	**11.464**	13.377	14.30%
EBONE(Europe)	**14.304**	20.624	30.65%
Sprintlink(US)	**9.732**	10.215	4.73%
Tiscali	**11.16**	13.209	15.51%
Avg	**11.665**	14.356	18.75%

实验结果表明，在所有测试的网络拓扑中，使用 SyncGNN 的能显著降低响应时延。这一结果充分证明了 SyncGNN 提取的特征能够有效辅助策略网络进行算力网络资源调度。

这种提升的原因在于 SyncGNN 网络实现了算力节点和路径节点之间的特征融合。在选择算力节点时，SyncGNN 不仅考虑了算力节点的特征，还整合了路径节点的特征，从而更准确地反映了传输的影响。

4.3.2 两阶段微调策略(TSFT)的有效性

实验设置：

(1) 数据集：NAS-Bench-201 数据集。

训练阶段：构建了包含 18 种异构设备的初始算力节点空间，涵盖了 GPU、CPU 等多种硬件类型，包括 NVIDIA 1080ti、Titan X 等，为每个训练设备收集了 900 个数据。

新设备：考虑了未知设备和未知平台，包括与初始算力节点空间不同但属于相同硬件类别的设备（如 NVIDIA GPU Titan RTX、Intel CPU Xeon Gold 6226 等），以及 Raspi4、ASIC-Eyeriss、FPGA 等全新类别的设备。

指标：采用均方误差(MSE)衡量预测计算时延与实际计算时延之间的误差，MSE 越低表示预测模型的性能越好。MSE 的计算公式如下：

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 \tag{16}$$

其中，y_i 表示实际计算时延，$\hat{y}_i$ 表示预测计算时延，n 表示数据样本的总数。MSE 的值域为非负实数，其中 0 表示预测完全准确，数值越大表示预测误差越大。

(2)基线

①FLOPS [38]：通过预测浮点运算来近似预测计算时延。

②Layer-wise[39]：通过测量每个操作的时延并求和来预测整个神经网络的时延。未考虑优化策略的影响。

③BRP-NAS[40]：采用端到端训练方法，需要大量测量数据。

表 5 显示了新设备和平台上的计算时延预测的比较结果，本文的方法(TSFT)在平均 MSE 上实现了 0.00304 的效果，相较于所有基线方法最优。这充分证明了本文提出的两阶段微调策略(TSFT)在提升新设备的计算时延预测模型性能方面的有效性。

为了评估两阶段微调策略(TSFT)带来的时间开销，分别计算在 5 个新设备上执行两阶段微调策略的时间，和端到端训练的 BRP-NAS 进行对比，结果如表 6 所示。

表 5　NAS-BENCH-201 上新设备的时延预测效果比较

Method	titan_rtx_256	gold_6226	fpga	raspi4	eyeriss	average
FLOPS	0.083	0.069	0.106	0.051	0.14	0.0898
Layer-wise	0.058	0.072	0.064	0.251	0.094	0.1078
BRP-NAS	0.0061	0.0023	**0.0012**	0.0146	**0.0014**	0.00512
基础模型	0.011	0.006	0.006	0.020	0.066	0.0218
TSFT	**0.0005**	**0.0015**	0.0016	**0.0093**	0.0023	**0.00304**

表 6　NAS-BENCH-201 上新设备的训练时间比较

Method	titan_rtx_256	gold_6226	fpga	raspi4	eyeriss	average
BRP-NAS	4223.88	3897.73	4471.96	5608.79	1447.72	3930.016
TSFT	**9.28**	**10.01**	**7.35**	**8.52**	**8.12**	**8.656**

综上,在对五种不同设备进行的两阶段微调策略(TSFT)实验中,训练时间开销非常小,平均仅为 8.656 秒,和 BRP-NAS 对比显著降低了训练时间。这一结果表明,两阶段微调策略(TSFT)带来的时间开销非常小,在实际应用中具有很高的可行性,尤其是在开放算力网络环境中需要快速部署新设备的场景中。

4.4　参数敏感性

4.4.1　算力候选节点数量对响应时延的影响

为了探究算力候选节点数量对响应时延的影响,本文在四个网络中进行了测试,记录了不同算力候选节点下的平均响应时延。如表 7 所示随着算力候选节点数量的增加,响应时延整体呈现下降趋势。这表明,增加算力节点的选择范围,可以增加调度方案的多样性,从而更有可能找到优化的调度决策。

特别地,当选择 5 个节点时,性能已接近理想状态。这说明,即使不遍历全网所有节点,仅通过精选 5 个候选节点,就能实现高效的调度效果,验证了本文节点筛选策略的有效性。

尽管部分数据集显示响应时延有所波动,这可能与特定网络拓扑的复杂性有关,如节点间的连接方式和数据传输路径的变化。但总体趋势显示响应时延逐渐降低,证实了节点筛选策略的有效性,表明通过选择有限数量的候选节点,就能达到高效的调度性能。

表 7　不同算力候选节点下任务的响应时延

网络拓扑	Node_3	Node_4	Node_5	Node_6	Node_7
Abilene	12.16	11.40	11.46	—	—
EBONE	17.86	15.36	14.30	13.59	13.47
Sprintlink	9.80	9.54	9.73	9.24	9.59
Tiscali	11.65	11.52	11.16	11.23	11.17

4.4.2　强化学习超参数设置

在强化学习算法中,超参数是影响模型训练效果和收敛速度的关键因素。本文针对不同的超参数进行了系统的实验研究,并依据实验结果确定其取值范围。针对每组超参数,在 Abilene 网络中进行训练,随后利用 50 个任务的平均响应时延进行评估。鉴于不同网络场景下结论具有一致性,在此仅介绍 Abilene 网络的实验结果。

对于策略网络学习率,分别测试了学习率取值为 0.1、0.01 和 0.001 这三个取值。实验结果如表 8 所示,当学习率为 0.1 时,实验结果值为 9.78,优于学习率为 0.01 和 0.001 的情况。这表明学习率为 0.1 时能够使策略网络在训练过程中更快收敛,同时避免因学习率过小导致训练过程过于缓慢或陷入局部最优解的问题。

表 8　不同策略网络学习率的平均响应时延

策略网络学习率	0.1	0.01	0.001
平均响应时延	9.78	11.38	11.40

对于价值网络学习率,同样测试了 0.1、0.01 和 0.001 三个取值。实验结果如表 9 显示,当学习率为 0.1 时,实验结果值为 7.73,优于另外两个取值。这说明在价值网络训练中,学习率为 0.1 能够更高效地评估状态价值。

表 9　不同价值网络学习率的平均响应时延

价值网络学习率	0.1	0.01	0.001
平均响应时延	7.73	9.78	11.51

对于衰减因子 γ,起到平衡当前奖励与未来奖励的作用。具体而言,较大的值 γ 意味着系统更注重未来的奖励,而较小的 γ 值则更偏向于当前奖励。本文对 γ 取值为 0.9、0.95 和 0.99 的情况进行了测试。实验结果如表 10 显示,当 γ 为 0.99 时,实验结果值为 7.73,优于 γ 为 0.9 和 0.95 的情况。这说明在本实验场景下,较大的衰减因子能够更有效地平衡当前奖励与未来奖励,从而获得更优的结果。

关于训练时间，为保证在每个拓扑下模型得到充分训练，episode 设置为 200000，具体训练时间如表 11 所示。需要注意的是，这种计算成本是离线产生的，并且可以根据环境的稳定性，不频繁地进行更新。

表 10 不同衰减因子的平均响应时延

衰减因子	0.90	0.95	0.99
平均响应时延	11.06	9.78	7.73

表 11 不同网络模型的训练时间

网络拓扑	训练时间(h)
Abilene	14.66
EBONE	15.15
Sprintlink	10.53
Tiscali	14.47
UsCarrier	18.56
Kdl	118.91

4.5 大规模算力网络的可扩展性

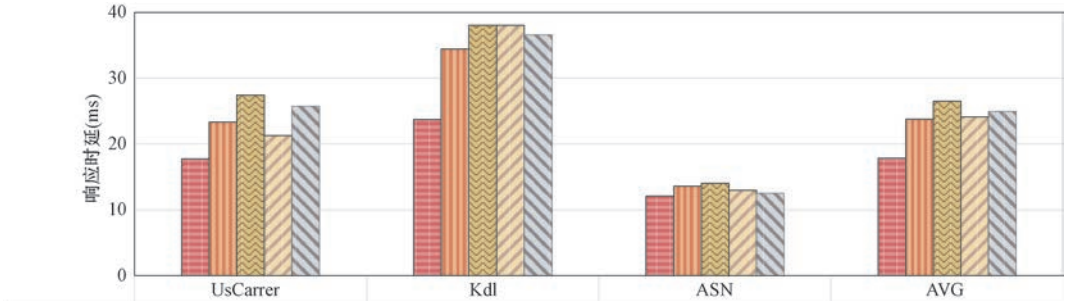
在本研究中，本文特别关注了算力网络资源调度系统在面对大规模算力网络时的可扩展性。考虑到现实世界中的大型服务提供商，如阿里云，运营的网络规模通常达到数百万级别的节点和链路，调度系统必须具备可扩展性。

为验证系统的可扩展性，在大型算力网络拓扑中进行了测试，选取了互联网 Topology Zoo 的三种网络拓扑：UsCarrier，Kdl[58] 和 ASN[59]。表 12 展示了这些网络拓扑的节点和链路的数量：

表 12 网络拓扑信息

网络拓扑	节点数量	链路数量
UsCarrier	158	378
Kdl	754	1790
ASN	1739	8558

在这些大型网络中，资源调度的复杂性随着网络规模的扩大而显著增加。本文测试了 100 个任务的平均响应时延，结果如图 19 所示：



	UsCarrer	Kdl	ASN	AVG
RLMS_CFN	17.69864436	23.7197069	12.04537219	17.82124115
Base1	23.30534328	34.42675292	13.58861387	23.77357002
Base2	27.44157703	38.00517974	14.03965420	26.49547032
Base3	21.28357837	38.00517974	12.96491941	24.08455917
Base4	25.71641528	36.56455781	12.48413533	24.92170281

图 19 各个网络拓扑的平均响应时延

RLMS-CPN 相对于各个基线的提升水平如表 13 所示：

表 13 相对基线提升水平

网络拓扑	Base1	Base2	Base3	Base4
UsCarrier	24.058%	35.504%	16.844%	31.178%
Kdl	31.101%	37.588%	37.588%	35.129%
ASN	11.357%	14.205%	7.093%	3.515%
AVG	25.038%	36.739%	26.006%	28.491%

实验数据表明，RLMS-CPN 在这些大型网络拓扑中取得了良好的性能提升。平均来看，RLMS-CPN 在大型算力网络中的响应时延明显低于基线方法，这表明 RLMS-CPN 能够有效处理大规模、复杂的网络资源调度问题。

这是由于算力节点筛选方案通过合理筛选候选节点，可以在不牺牲性能的前提下降低计算复杂性，实现高效的资源调度，这对于处理大量时延敏感型任务的大型云服务提供商来说，具有参考价值。

4.6 调度方案的计算时间开销

为了全面评估方法引入的时间开销，对四种不同网络拓扑结构进行了测试，每个拓扑结构均测试了 100 个任务。各网络拓扑下任务方案的计算时间汇总于表 14 中。

表 14 不同网络拓扑方案的计算时间(单位：秒)

网络拓扑	Abilene	EBONE	Sprintlink	Tiscali	Avg
计算时间	0.0031	0.0050	0.0048	0.0052	0.0045

根据上表数据，RLMS-CPN 在所有测试网络拓扑中的平均计算时间为 0.0045 秒。尽管引入了这一微小的时间开销，RLMS-CPN 能够有效地将响应时延平均降低了 22.61%，显示出其开销与效益之

间的合理平衡。

4.7　任务类型对调度性能的影响

为了评估方法在不同类型任务下的适应性,我们进一步将任务细分为 IO 密集型、计算密集型和混合型,并分别进行实验对比。

IO 密集型任务:以高数据传输需求为主,核心在于频繁且大量的数据交换。在实验中,我们设计了包含 50 个任务的测试场景,每个任务进行约 10 次数据传输,每次传输的数据量分别为 100 KB、1 MB 和 10 MB。这种任务的性能主要受限于网络传输效率,因此优化数据传输路径和降低传输时延是关键目标。

计算密集型任务:以高计算资源需求为主,任务的核心在于复杂的计算操作。在我们的实验中,计算密集型任务通过选择神经网络模型更加复杂的 50 个任务来体现,这些任务的平均模型参数大小为 1.20312 MB。这种任务的性能主要受限于计算资源的分配和利用效率,因此优化计算资源的调度和分配是关键目标。

混合型任务:兼具计算密集型任务和 IO 密集型任务的特点,既包含大量的数据传输,又涉及复杂的计算操作。其调度策略需要同时优化数据传输阶段和计算阶段,以实现整体性能的提升。混合型任务的评估结果见 4.2 节分析。

实验结果表明,RLMS-CPN 方法在不同类型的任务调度中均展现出显著优势。对于 IO 密集型任务,实验结果见表 15,随着数据量从 100 KB 增加到 10MB,RLMS-CPN 始终保持最优性能,验证了其在网络传输路径优化方面的有效性。在计算密集型任务,实验结果见表 16,RLMS-CPN 的响应时延优于所有对比方案,充分体现了其在计算资源调度上的先进性。

综合 4.2 节对混合型任务的分析,可以得出结论:RLMS-CPN 通过协同优化传输路径选择和计算资源分配,能够有效适应不同类型的任务需求,尤其在任务复杂度增加(如数据量增大或计算需求提高)时,展现出稳定的性能优势。

5　结　论

本研究首次提出了一个面向 AI 任务的算力资源及网络资源协同调度系统(RLMS-CPN),旨在通过智能算法实现资源的联合优化,以满足 AI 任务的低响应时延需求。

RLMS-CPN 核心在于基于 GNN 的强化学习算网联合调度算法。SyncGNN 结构充分利用图神经网络的优势,通过信息交换进行计算特征和网络特征的有效融合,协同优化算力节点选择和路径规划。针对开放网络环境下的计算时延估计问题,本文提出的两阶段微调策略(TSFT)显著提升了模型对新设备的计算时延的估计精度。

通过 NS3 的大规模仿真评估表明,与现有的调度策略相比,本文的策略在降低响应时延方面具有显著优势,平均减少了 22.61% 的响应时延,且在大规模网络中保持了良好的可扩展性。

表 15　IO 密集型任务的响应时延

传输数据量	RLMS-CPN	Base1	Base2	Base3	Base4	GNN+DQN
100KB	**105.10**	145.99	145.06	132.54	141.78	140.04
1MB	**147.87**	187.50	184.33	175.95	181.26	190.91
10MB	**600.34**	661.40	633.42	654.62	608.81	754.98

表 16　计算密集型任务的响应时延

RLMS-CPN	Base1	Base2	Base3	Base4	GNN+DQN
35.749	40.430	39.075	39.455	38.005	45.957

参　考　文　献

[1] Lei Bo, Liu Zeng-Yi, Wang Xu-Liang, et al. A new edge computing solution based on cloud, network, and edge integration: Computing power network. Telecommunication Science, 2019, 35(9): 44-51 (in Chinese)
(雷波, 刘增义, 王旭亮等. 基于云、网、边融合的边缘计算新方案:算力网络. 电信科学, 2019, 35(9): 44-51)

[2] Jia Qing-Min, Ding Rui, Liu Hui, et al. A review of research progress on computing power network. Journal of Network and Information Security, 2021, 7(5): 1-12 (in Chinese)
(贾庆民, 丁瑞, 刘辉等. 算力网络研究进展综述. 网络与信息安全学报, 2021, 7(5): 1-12)

[3] He Tao, Yang Zhen-Dong, Cao Chang, et al. Analysis of sev-

eral key technical issues in the development of computing power network. Telecommunication Science, 2022, 38(6): 62-70 (in Chinese)
(何涛, 杨振东, 曹畅等. 算力网络发展中的若干关键技术问题分析. 电信科学, 2022, 38(6): 62-70)

[4] Zhou Xu, Li Zhuo. Cloud-edge-end collaborative scheduling technology for computing power network. ZTE Communications, 2023, 29(4): 32-37 (in Chinese)
(周旭,李琢. 面向算力网络的云边端协同调度技术. 中兴通讯技术,2023,29(4): 32-37)

[5] Elbamby M S, Perfecto C, Bennis M, et al. Toward low-latency and ultra-reliable virtual reality. IEEE Network, 2018, 32(2): 78-84

[6] Peng Y, Shi B, Jiang T, et al. A survey on in-vehicle time-sensitive networking. IEEE Internet of Things Journal, 2023, 10(16): 14375-14396

[7] Xu S, Perez M, Yang K, et al. Determination of the latency effects on surgical performance and the acceptable latency levels in telesurgery using the dV-Trainer® simulator. Surgical Endoscopy, 2014, 28(9): 2569-2576

[8] Han Xue-Ying. Research and simulation verification of resource allocation methods for computing-network convergence [Master's thesis]. Beijing University of Posts and Telecommunications, Beijing, 2023 (in Chinese)
(韩雪莹. 面向算网融合的资源分配方法的研究与仿真验证[硕士学位论文]. 北京邮电大学, 北京, 2023)

[9] Zhu Si-Kai. Research on intelligent scheduling mechanisms for computing-network convergence systems [Master's thesis]. Guangdong Polytechnic Normal University, Guangzhou, 2023 (in Chinese)
(朱思锴. 面向算网融合系统的智能调度机制研究[硕士学位论文]. 广东技术师范大学, 广州, 2023)

[10] Gong Xiao-Min. Research on software-defined computing power perception and routing algorithms [Master's thesis]. Academy of Military Sciences, Beijing, 2024 (in Chinese)
(龚晓敏. 软件定义算力感知与路由算法研究[硕士学位论文]. 军事科学院, 北京, 2024)

[11] Liang Zi-Luo. Research on routing and resource allocation technologies in computing-network integrated optical networks [Master's thesis]. Beijing University of Posts and Telecommunications, Beijing, 2024 (in Chinese)
(梁滋洛. 算网融合光网络路由与资源分配技术研究[硕士学位论文]. 北京邮电大学, 北京, 2024)

[12] Hu T C. Multi-commodity network flows. Operations Research, 1963, 11(3): 344-360

[13] Gurobi Optimization LLC. Gurobi optimizer reference manual. Houston: Gurobi Optimization LLC, 2021

[14] Namyar P, Arzani B, Beckett R, et al. Minding the gap between fast heuristics and their optimal counterparts//Proceedings of the 21st ACM Workshop on Hot Topics in Networks. Austin, USA, 2022: 138-144

[15] Zhang J, Xi K, Zhang L, et al. Optimizing network performance using weighted multipath routing//Proceedings of the 21st International Conference on Computer Communications and Networks. Munich, Germany, 2012: 1-7

[16] Rusek K, Almasan P, Suárez-Varela J, et al. Fast traffic engineering by gradient descent with learned differentiable routing//Proceedings of the 18th International Conference on Network and Service Management. Thessaloniki, Greece, 2022: 359-363

[17] Pei X, Sun P, Hu Y, et al. Enabling efficient routing for traffic engineering in SDN with deep reinforcement learning. Computer Networks, 2024, 241: 110220

[18] Lin B, Guo Y, Luo H, et al. TITE: a transformer-based deep reinforcement learning approach for traffic engineering in hybrid SDN with dynamic traffic. Future Generation Computer Systems, 2024, 161: 95-105

[19] Sun P, Lan J, Li J, et al. A scalable deep reinforcement learning approach for traffic engineering based on link control. IEEE Communications Letters, 2020, 25(1): 171-175

[20] Sun P, Lan J, Guo Z, et al. Improving the scalability of deep reinforcement learning-based routing with control on partial nodes//Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing. Barcelona, Spain, 2020: 3557-3561

[21] Zhang J, Ye M, Guo Z, et al. CFR-RL: traffic engineering with reinforcement learning in SDN. IEEE Journal on Selected Areas in Communications, 2020, 38(10): 2249-2259

[22] Ye M, Zhang J, Guo Z, et al. DATE: disturbance-aware traffic engineering with reinforcement learning in software-defined networks//Proceedings of the 29th International Symposium on Quality of Service. Hangzhou, China, 2021: 1-10

[23] Ye M, Zhang J, Guo Z, et al. FlexDATE: flexible and disturbance-aware traffic engineering with reinforcement learning in software-defined networks. IEEE/ACM Transactions on Networking, 2022, 31(4): 1433-1448

[24] Ye M, Hu Y, Zhang J, et al. Mitigating routing update overhead for traffic engineering by combining destination-based routing with reinforcement learning. IEEE Journal on Selected Areas in Communications, 2022, 40(9): 2662-2677

[25] Xu Z, Yan F Y, Singh R, et al. TEAL: learning-accelerated optimization of WAN traffic engineering//Proceedings of the ACM Special Interest Group on Data Communication Conference Conference. New York, USA, 2023: 378-393

[26] Bernárdez G, Suárez-Varela J, López A, et al. Magnneto: a graph neural network-based multi-agent system for traffic engineering. IEEE Transactions on Cognitive Communications and Networking, 2023, 9(2): 494-506

[27] Guo Y, Lin B, Tang Q, et al. Distributed traffic engineering in hybrid software defined networks: a multi-agent reinforcement learning framework. IEEE Transactions on Network and Service Management, 2024, 21(6) 6759-6769

[28] Gómez-DelaHiz J, Galán-Jiménez J. Improving the traffic en-

gineering of SDN networks by using local multi-agent deep reinforcement learning//Proceedings of the IEEE Network Operations and Management Symposium. Seoul, Republic of Korea, 2024: 1-5

[29] Luan Z, Li Q, Jiang Y, et al. MATE: when multi-agent deep reinforcement learning meets traffic engineering in multi-domain networks. Computer Networks, 2024, 247: 110399

[30] Nguyen D L, Nguyen P L, Ji Y. Traffic engineering in large-scale networks via multi-agent deep reinforcement learning with joint training//Proceedings of the 33rd International Conference on Computer Communications and Networks. Hawaii, USA, 2024

[31] Long Y, Wang Z, Lan S, et al. Energy-latency tradeoff for task offloading and resource allocation in vehicular edge computing. Computer Networks, 2025, 258: 111026

[32] Li X, Chen L, Yuan Z, et al. AIHO: enhancing task offloading and reducing latency in serverless multi-edge-to-cloud systems. Future Generation Computer Systems, 2025, 165: 107607

[33] Yang W, Liu Z, Liu X, et al. Deep reinforcement learning-based low latency task offloading for mobile-edge computing networks. Applied Soft Computing, 2024, 166: 112164

[34] Kim M, Jang J, Choi Y, et al. Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks. IEEE Transactions on Mobile Computing, 2024,23(12): 15149-15166

[35] Yang B, Cao X, Bassey J, et al. Computation offloading in multi-access edge computing: a multi-task learning approach. IEEE Transactions on Mobile Computing, 2020, 20(9): 2745-2762

[36] Cong Y, Xue K, Wang C, et al. Latency-energy joint optimization for task offloading and resource allocation in MEC-assisted vehicular networks. IEEE Transactions on Vehicular Technology, 2023, 72(12): 16369-16381

[37] Jin W, Liu G, Gu H. MEC multi-objective task offloading algorithm for joint energy and latency optimization//Proceedings of the IEEE 10th International Conference on High Performance and Smart Computing. New York, USA, 2024: 43-48

[38] Wang H, Wu Z, Liu Z, et al. Hat: Hardware-aware transformers for efficient natural language processing. arXiv: 2005.14187, 2020

[39] Cai H, Zhu L, Han S. Proxylessnas: Direct neural architecture search on target task and hardware. arXiv: 1812.00332, 2018

[40] Dudziak L, Chau T, Abdelfattah M, et al. Brp-nas: Prediction-based nas using gcns//Proceedings of the Advances in Neural Information Processing Systems. Virtual, 2020: 10480-10490

[41] Lee H, Lee S, Chong S, et al. Hardware-adaptive efficient latency prediction for nas via meta-learning//Proceedings of the Advances in Neural Information Processing Systems. Virtual, 2021: 27016-27028

[42] Akhauri Y, Abdelfattah M S. Multi-predict: Few shot predictors for efficient neural architecture search. arXiv: 2306.02459, 2023

[43] Yu F, Zhang M, Dong H, et al. Dast: Unsupervised domain adaptation in semantic segmentation based on discriminator attention and self-training//Proceedings of the AAAI Conference on Artificial Intelligence. Virtual, 2021: 10754-10762

[44] Sanchez-Lengeling B, Reif E, Pearce A, et al. A gentle introduction to graph neural networks. Distill, 2021, 6(9): e33

[45] Zhu H, Gupta V, Ahuja S S, et al. Network planning with deep reinforcement learning//Proceedings of the ACM Special Interest Group on Data Communication Conference. Virtual, 2021: 258-271

[46] Fan W, Ma Y, Li Q, et al. Graph neural networks for social recommendation//Proceedings of the World Wide Web Conference. San Francisco, USA, 2019: 417-426

[47] Mohamed A, Qian K, Elhoseiny M, et al. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Seattle, USA, 2020: 14424-14432

[48] Lange O, Perez L. Traffic prediction with advanced graph neural networks. DeepMind Research Blog, 2020

[49] Hamilton W L, Ying R, Leskovec J. Representation learning on graphs: Methods and applications. arXiv: 1709.05584, 2017

[50] Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry//Proceedings of the International Conference on Machine Learning. Sydney, Australia, 2017: 1263-1272

[51] NS-3 Consortium. Network Simulator 3 (NS-3) v3.38 [CP/OL]. 2023. https://www.nsnam.org/ (accessed 2024-06-20)

[52] Vanini E, Pan R, Alizadeh M, et al. Let it flow: Resilient asymmetric load balancing with flowlet switching//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation. Boston, USA, 2017: 407-420

[53] Konda V, Tsitsiklis J. Actor-critic algorithms//Proceedings of the Advances in Neural Information Processing Systems. Denver, USA, 1999: 1008-1014

[54] Zhang Y. Abilene Network Traffic Dataset [DB/OL]. 2019. http://www.cs.utexas.edu/yzhang/research/AbileneTM/ (accessed 2019-04-22)

[55] Spring N, Mahajan R, Wetherall D. Measuring ISP topologies with Rocketfuel//Proceedings of the ACM Special Interest Group on Data Communication Conference. Pittsburgh, USA, 2002: 133-145

[56] Zhang J, Li X, Chen L, et al. Scheduling workflows with limited budget to cloud server and serverless resources. IEEE

Transactions on Services Computing, 2023, 17(4): 1766-1779

[57] Zhu J, Li X, Ruiz R, et al. Scheduling stochastic multi-stage jobs to elastic hybrid cloud resources. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(6): 1401-1415

[58] Knight S, Nguyen H X, Falkner N, et al. The internet topology zoo. IEEE Journal on Selected Areas in Communications, 2011, 29(9): 1765-1775

[59] The CAIDA AS Relationships Dataset, 2022 (Accessed: 2022)

WANG Jing, M. S. candidate. Her research interests include computer networks and computing power network.

XIE Kun, Ph. D., professor. Her research interests include computer networks, network security, computing power networks, and artificial intelligence.

ZENG Xin, Ph. D. candidate. His research interests include network measurement and in-band telemetry.

ZHAO Peng-Cheng, M. S. candidate. His research interests include network security and computing power networks.

WEN Ji-Gang, Ph. D., associate professor. His research interests include network security and computing power networks.

XIE Gao-Gang, Ph. D., professor. His research interests include network architecture and network measurement.

Background

The problem studied in this paper belongs to the resource scheduling related issues in the field of computing power network. Specifically, it focuses on how to solve the decoupling problem of computing resource and network resource allocation in the computing power network for latency-sensitive tasks, so as to achieve collaborative scheduling and meet the low-latency requirements of latency-sensitive tasks. This paper mainly involves three areas: Traffic engineering mainly focuses on path planning, aiming to improve network performance by optimizing routing strategies, and is divided into link-level and path-level traffic engineering. Among them, the linear programming method can provide the optimal solution, but the computational cost is high when the network scale increases. Although the heuristic algorithm reduces the complexity, it is not flexible enough. Although the traffic engineering based on reinforcement learning has potential, it faces challenges such as dimension explosion and increased computational cost. Currently, there is no work to achieve a collaborative scheduling system for computing power resources and network resources for AI tasks. The core goal of task offloading is to select an appropriate computing power execution node for computing tasks. However, the existing research has deficiencies in the selection of computing power nodes. Many methods do not consider the computing characteristics of AI tasks, resulting in inaccurate estimation of computing latency. Moreover, they only focus on node selection and ignore the impact of path planning on the overall performance. For computing latency prediction, the early models estimated computing latency based on FLOPS, with largeerrors. Although there have been improvements later, when dealing with a large number of tasks and devices, they still face the dual challenges of data efficiency and generalization ability, and have failed to solve the problem of predicting the computing latency of tasks on new devices in an open network environment.

In response to the above problems, this paper proposes a joint scheduling optimization framework. Based on reinforcement learning, a joint scheduling optimization framework for computing power network is proposed, which can collaboratively optimize the selection of computing power nodes and path planning and minimize the response latency of tasks. Experiments on six datasets show that compared with the existing methods, the response latency is reduced by an average of 22.61%. A computing latency estimation model is designed. For the problem of computing latency estimation of new devices in an open space, a two-stage fine-tuning strategy model based on neural network is proposed. With only 50 new samples, a low prediction error of 0.003 is achieved, effectively improving the accuracy of computing latency prediction for new devices. A new graph neural network structure is constructed. A new graph neural network structure SyncGNN is designed, which effectively integrates computing resources and network transmission resources, realizes feature fusion, and collaboratively optimizes the selection of computing power nodes and path planning. Compared with the traditional multi-layer perceptron, the latency performance is improved by 18.75%.