XXXX

DS-PPO：面向高并发CNN模型推理的云边端异构硬件资源协同调度

张阔[1]，闫亚旗[1]，安颖[1]，董玉池[1]，张民贵[1]，袁刚[2]
（1. 中国铁塔股份有限公司 北京 100089；
2. 中移（苏州）软件技术有限公司 苏州 215000）

摘　要：随着人工智能物联网（AIoT）技术的发展和硬件性能的提升，云边端协同场景下引入卷积神经网络（CNN）模型进行推理已成为趋势，以满足智能AI应用对实时性的需求。针对异构计算资源（如CPU、GPU、NPU、FPGA）的高效利用问题，提出了一种创新的任务调度系统，以支持高并发CNN模型推理。该系统采用DeepSets模型进行任务和环境特征提取，并提升近端策略优化算法的泛化能力实现任务调度决策，确保在复杂多变的环境中实现精准高效的任务分配。系统已在Kubernetes和Kosmos平台上部署，并通过高性能服务器、NVIDIA Jetson Xavier NX和NVIDIA Jetson Nano等异构处理单元进行了验证。实验结果显示，相较于现有调度算法，该系统的性能提升了275.93%，资源利用率提高38.86%，显著提高了云边端计算的效率和可靠性，为智能AI应用提供了坚实的支持。

关键词：云边端协同；高并发；CNN；DeepSets；PPO泛化；异构资源；任务调度
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.

DS-PPO: Cloud-Edge-End Collaborative Scheduling of Heterogeneous Hardware Resources for Highly Concurrent CNN Model Inference

ZHANG Kuo[1], YAN Yaqi[1], AN Ying[1], DONG Yuchi[1], ZHANG Mingui[1], YUAN Gang[2]
1. China Tower Corporation Limited, Beijing, 100089
2. China Mobile (Suzhou) Software Technology Co., Ltd., Suzhou 215000

Abstract: With the advancement of artificial intelligence of things (AIoT) technology and the enhancement of hardware performance, convolutional neural network (CNN) models were increasingly introduced for inference in collaborative cloud-edge-end scenarios to satisfy real-time requirements of intelligent AI applications. To address the efficient utilization of heterogeneous computing resources including CPUs, GPUs, NPUs, and FPGAs, an innovative task scheduling system was proposed to support high-concurrency CNN model inference. The DeepSets model was em-

收稿日期：2025−12−16；修回日期：2025−02−12
通信作者：张阔，zhangkuo3@chinatowercom.cn

ployed to extract features from tasks and environmental constraints, while the generalization capability of the Proximal Policy Optimization (PPO) algorithm was enhanced to achieve task scheduling decisions. Precise and efficient task allocation was ensured in complex dynamic environments through this approach. The proposed system was deployed on Kubernetes and Kosmos platforms, with comprehensive validation conducted across various heterogeneous processing units, including high-performance servers, NVIDIA Jetson Xavier NX, and NVIDIA Jetson Nano devices. Experimental results demonstrate that compared to existing scheduling algorithms, the proposed system improves performance by 275.93% and increases resource utilization by 38.86%, significantly enhancing the efficiency and reliability of cloud-edge computing frameworks and providing robust support for intelligent AI applications.

Key words: Cloud-edge-end collaboration, high concurrency, CNN, DeepSets, PPO generalization, heterogeneous resources, task scheduling

1 引言

在工业物联网（IoT）和5G技术的推动下，云边端（Cloud-Edge-End, CEE）协同计算框架的兴起显著提升了计算服务能力，尤其在人工智能应用的效率和用户体验方面表现突出[1]。然而，该框架的固有复杂性也给系统设计带来了重大挑战，尤其是任务调度问题——如何在异构硬件架构与计算能力之间实现系统性能的最大化[2]。其中，计算资源的异构性是核心难题之一：通用处理器与专用AI加速器等多种硬件架构并存，加之任务需求的多样性，使得资源管理变得异常复杂[3]。

任务调度的目标是为任务匹配合适的资源，以优化系统性能。然而在云边端协同的异构环境中，现有方法往往难以全面考虑任务的多样化需求，特别是在对设备亲和性要求较高的AI应用中[4]。Attiya等人[5]在云雾混合范式中提出基于改进人造生态系统优化的任务调度方案，以降低物联网应用请求的延迟。Alsaidy等人[6]通过启发式算法改进粒子群优化（PSO）实现云计算任务调度。Wang等人[7]结合强化学习设计制造系统的状态与动作空间，以优化生产调度的利润、效率及功耗。这些工作均针对特定计算环境进行优化，但难以完全应对AI任务的复杂性[8]。

目前，主流研究对计算资源异构性的考量仍不充分，尤其缺乏对CPU、GPU、NPU等不同计算单元的系统性建模。Yuan等人[9]基于HGIC的多队列调度实现混合云场景下的异构任务管理；Zhou[10]在异构多处理器系统中进行高效任务调度以优化能耗与安全质量；Houssein等人[1]系统综述了云雾范式下的任务调度技术，强调了该领域研究的紧迫性。设计和实现能够管理大规模高性能服务器集群的调度系统既具挑战性又十分迫切。当前以Kubernetes为代表的云平台主要面向单集群管理，并不天然支持云边端多集群场景下的复杂任务分配与资源调度，对GPU、NPU等多样化计算资源的支持也较为有限，制约了云边端协同场景下计算任务的高效执行。

云边端协同计算框架需综合考虑服务响应时间与整体资源利用率，其核心调度挑战在于如何在庞大的决策空间中进行在线决策，并兼顾时间效率与资源优化。为此，本文提出一种基于强化学习的云边端协同实时调度策略，以应对任务与计算资源的异构性。该方法通过DeepSets神经网络优化PPO算法的泛化能力，并结合任务突发情况设计智能队列进行优先级排序，在考虑异构计算能力及网络影响的条件下，保障任务的高效分配与整体资源利用率的最大化。

本文在生产测试域环境中，基于高性能服务器、NVIDIA Jetson Xavier NX和NVIDIA Jetson Nano等设备实现了云边端计算框架与调度模块，

系统基于 Kubernetes 和 Kosmos 平台构建。实验表明，本文所提算法优于主流任务调度方法，尤其在任务突发场景下，性能提升达 275.93%，资源利用率提升 38.86%。

本研究主要贡献如下：

（1）针对任务分配对应的离散决策空间与资源利用对应的连续决策空间，提出基于 DeepSets 神经网络的任务与环境特征提取方法，优化近端策略优化（PPO）的泛化能力，以适应云边端协同场景中的异构资源，显著降低任务完成时间与等待时间，并提高计算资源利用率。

（2）基于扩展的 Kubernetes 和 Kosmos 平台实现了一套云边端协同计算系统框架，并将所提 DS-PPO 算法集成于调度模块，提升了资源与任务的高效匹配能力。

（3）通过生产测试域的平台实验，验证了在不同任务到达率场景下，DS-PPO 模型在云边端协同计算框架中优化算法性能与资源利用率方面的优越性。

2　相关工作

2.1　云边端协同计算框架

Kubernetes 作为主流的容器编排平台，凭借其高性能且公平的调度器被广泛应用于资源监控与 CNN 模型推理任务调度中。然而，在云边端协同应用场景下，Kubernetes 存在两个关键限制[11]：

（1）单集群调度局限：Kubernetes 专为单集群编排设计，适用于云或边缘服务器集群的独立管理。云边端协同场景涉及跨云、边、端的多个异构集群，需要实现任务与资源的跨集群调度与统一监控，这一需求超出了 Kubernetes 的默认能力范围。

（2）异构资源支持有限：Kubernetes 默认调度器主要支持 CPU 资源的分配，尽管可通过设备插件（Device Plugins）扩展对 GPU 的支持，但对于 NPU、TPU、FPGA 等多样化的异构计算单元，其资源抽象与调度机制仍显不足，难以实现细粒度的资源管理与协调。

为弥补上述不足，本文采用自研的 Kosmos 平台作为云边端协同场景下的跨集群统一编排与调度框架。Kosmos 平台在 Kubernetes 原生能力基础上进行了以下关键扩展与增强：

（1）跨集群统一编排：Kosmos 通过中心化的控制平面实现对多个 Kubernetes 集群的协同管理，支持跨集群的应用部署、服务发现与流量调度。其采用声明式 API 与策略驱动机制，允许用户以统一视角管理分布在云、边、端的计算资源，并实现任务的智能分发与迁移。

（2）异构资源统一抽象与发现：Kosmos 平台通过节点驱动接口（如 nvidia-smi、npu-smi）与 GRPC 通信协议实时收集各节点的异构资源状态，包括 GPU、NPU、FPGA 等加速器的算力、内存与利用率信息。平台将这些异构资源统一抽象为可调度的扩展资源，为上层调度器提供完整的资源视图。

（3）资源感知调度扩展：Kosmos 提供了可插拔的调度器扩展接口，允许集成自定义调度策略。本文基于此接口实现了 DS-PPO 调度模块，使其能够依据实时资源状态与任务需求进行智能决策，并将调度结果通过 Kosmos API 下发至对应集群执行。

（4）网络与存储协同：针对云边端场景下的跨集群通信与数据访问需求，Kosmos 提供了虚拟网络覆盖与统一存储编排能力，降低了跨域数据传输延迟，保障了分布式任务执行的效率与一致性。

通过 Kubernetes 与 Kosmos 的协同，本文构建了一个既保留 Kubernetes 成熟生态、又具备跨集群管理与异构资源调度能力的云边端协同计算框架，为后续智能调度算法的实现提供了系统基础。

2.2 云边端协同异构任务调度

在云边端协同场景下，除了构建跨集群任务调度与异构资源监控框架外，设计高效调度算法是完成计算任务的关键。目前，针对云计算、边缘计算等场景已有多种调度算法。Narayanan等人[12]提出异构感知调度程序Gavel，将调度策略建模为优化问题，利用不同加速器在不同任务中的差异来分配。Feng等人[13]提出基于异构移动架构的计算和资源分配框架，保证联邦学习有效实施并通过多维控制优化能耗与能量收集。Zhong等人[14]通过资源优化和弹性实例定价提升异构任务分配成本效益，适合负载动态变化的云场景。

上述方法主要针对任务与设备异构问题，使用优化算法与模型实现实时调度。然而，云边端协同场景中，异构性不仅存在于计算芯片与任务之间，还体现在云、边、端设备整体计算能力的差异，如图1所示。

目前这方面研究相对不足，如图2所示，移动云智算平台对100个CNN模型推理测试表明，不同类型计算芯片在相同AI任务上存在显著时延差异，若调度不当会导致资源浪费和整体利用率偏低。本文在云边端协同架构内，通过强化学习优化的策略算法，实现异构任务与异构计算能力的统一调度。

2.3 强化学习调度

近年来研究表明，强化学习在调度算法性能提升方面具有显著效果[15]。RL通过智能体与环境交互，以奖励信号为反馈进行学习，已成功应用于游戏、机器人和资源管理等领域[16]。

现有RL在调度领域的应用主要集中于云计算和边缘计算，对云边端协同场景及其训练环境的研究仍较匮乏。同时，强化学习模型在实际应用中存在泛化能力不足的问题[17]，即在训练环境表现良好，在新环境或新CNN推理任务中性能下降。DeepSets[18]作为处理大小可变集合输入并具有置换不变性的模型，通过变换函数和聚合函数组合生成集合特征向量，为提升模型泛化能力提供了有效途径。

3 云边端协同系统框架

本文提出的云边端协同系统架构如图3所示，采用分层架构设计，顶层使用Kosmos高级容器管理平台来实现跨云边端连续体场景的统一管理编排，Kubernete负责在每个集群中处理容器的部署与管理任务，并且Kosmos通过提供访问控制和其他服务来简化集群的管理操作。

该分层设计实现了对多个Kubernetes集群的协同管理。在云、边、端各节点上均集成监控模

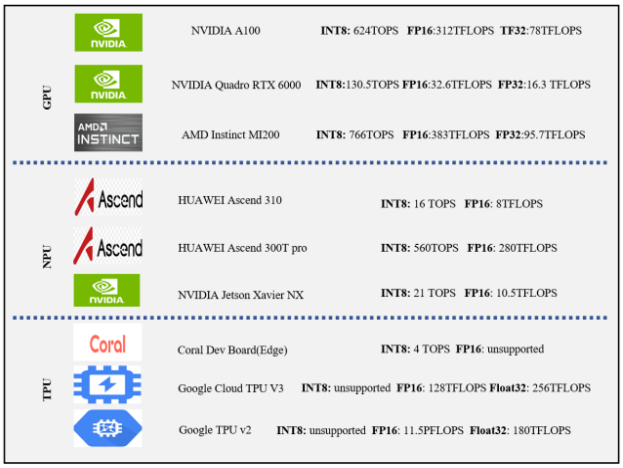


图1 异构性

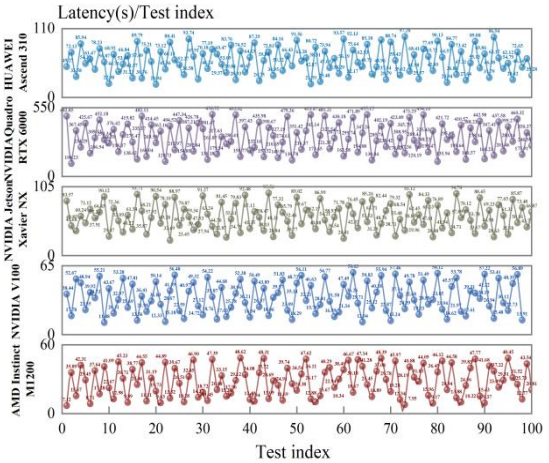


图2 不同设备上运行100个CNN模型的推理延迟

块，利用特定工具（如针对 NVIDIA Jetson Xavier NX 的 npu-smi）获取实时数据，并通过 GRPC 聚合到调度器。

为进一步优化资源管理，系统基于 Docker 为各类节点构建定制环境镜像，镜像按处理器/加速器类型定制。调度器通过 Kosmos API 拉取并启动相应镜像执行分布式任务，并以 JSON 格式分发任务，控制各集群各节点任务部署。该框架实现了高性能、资源感知的调度策略，有效提升云边端异构资源环境的整体性能和资源利用率。

本文提出的云边端协同系统框架基于扩展的 Kubernetes 与 Kosmos 平台实现跨集群的统一编排与调度，与当前主流的开源多集群管理工具（如 Karmada、OpenYurt 等）相比，其设计目标与能力侧重点存在以下差异：

（1）异构资源管理的深度扩展：Karmada 与 OpenYurt 主要面向多云/边缘场景下的多集群应用分发与运维，在资源抽象上仍以 CPU/内存等通用资源为主，对 GPU、NPU、FPGA 等异构计算单元的支持依赖厂商插件，缺乏统一的资源建模与调度策略优化机制。本文框架则通过 Kosmos 平台集成节点驱动（如 nvidia-smi、npu-smi）与 GRPC 通信层，实现对各类异构计算资源的实时感知与细粒度管控，并结合 DeepSets-PPO 调度算法进行资源-任务匹配优化，从而更好地适配高并发 CNN 推理任务对异构算力的差异化需求。

（2）调度策略的智能性与自适应性：现有多集群工具大多采用基于规则或负载均衡的静态调度策略，难以应对云边端场景下任务类型多样、资源动态变化等挑战。本文框架将强化学习调度器作为核心组件，能够根据实时环境状态自主决策任务分配策略，在降低任务完成时间的同时提升资源利用率，而 Karmada 等工具目前尚未集成此类智能调度能力。

（3）跨集群协同的任务粒度与网络优化：OpenYurt 注重边缘自治与云边协同的运维连续性，Karmada 强调多集群应用的策略一致性，二者在任务级调度与跨集群网络优化方面能力有限。本文框架则针对 CNN 推理任务的特点，设计了任务队列优先级调度与跨集群网络感知机制，能够在调度决策中综合考虑节点计算能力与集群间网络延迟，从而优化端到端任务执行效率。

综上所述，本文框架并非替代现有多集群管理工具，而是在其基础上进一步强化了异构资源统一抽象、智能调度决策与 CNN 任务优化支持三方面能力，为云边端协同场景下的高并发 AI 推理任务提供了更为专有的系统支持。

在所述云边端协同框架中，高并发 CNN 模型推理任务的调度可形式化为一个混合整数线性规划问题，其核心是在满足资源约束的前提下，将动态到达的任务合理分配到异构节点，以优化系统整体性能。下面对其进行数学建模：

设系统中共有 K 个集群，每个集群 k 包含 M_k 个节点，节点 n 的资源状态向量定义为 $\mathbf{R}_n=[c_{n1},c_{n2},\ldots,m_{n1},m_{n2},\ldots,s_n,b_n]$，分别表示不同类型计算单元（如 CPU、GPU、NPU）的可用核数、内存容量、存储空间和网络带宽。任务 j 的资源需求向量为 $D_j=[\hat{c}_{j1},\hat{c}_{j2},\ldots,\hat{m}_{j1},\hat{m}_{j2},\ldots,\hat{s}_j,\hat{t}_j]$，其中 $\hat{t}_j$ 表示预估执行时间。定义二元决策变量 $x_{j,n}\in\{0,1\}$ 表示任务 j 是否分配给节点 n，连续变量 C_j 表示任务 j 的完成时间，W_j 表示其等待时间。优化目标为最小化平均任务完成时间与等待时间，同时最大化资源利用率，可表述为：

$$Minimize \quad \frac{1}{N}\sum_{j=1}^{N}C_j+\frac{1}{N}\sum_{j=1}^{N}W_j-\frac{\eta}{M}\sum_{n=1}^{M}U_n, \quad (1)$$

其中 U_n 表示节点 n 的资源利用率，η 为权重系数。

约束条件主要包括以下几类：首先是任务分配唯一性约束，即每个任务必须被分配且仅被分配一次，数学表达为 $\sum_{n-1}^{m}x_{j,n}=1,\forall j$；其次是资源

容量约束，要求任意节点上分配的资源总量不得超过其可用资源，即 $\sum_{j=1}^{N}\mathbf{D}_j\cdot x_{j,n}\le\mathbf{R}_n,\ \forall n$；此外还有时间逻辑约束，确保任务完成时间不早于其到达时间、等待时间与执行时间之和，以及变量的非负性与整数性约束。

上述模型是一个典型的混合整数线性规划问题，其变量既包含整数分配决策，也包含连续时间变量，且目标函数与约束均为线性形式。由于问题规模随任务数、节点数和资源类型数呈指数增长，该问题属于NP-Hard类别，传统优化方法在动态高并发环境下难以实时求解。因此，本文转而采用基于深度强化学习的调度方法，通过智能体与环境的交互学习近似最优策略，在保证调度实时性的同时，实现系统性能的持续优化。

4 构建DS-PPO模型调度策略

本文采用强化学习代理作为调度器核心，实现高效CNN模型推理任务分配和资源管理。如图4所示，调度器的环境由CNN任务队列、系统时钟和云边端异构计算资源构成。任务队列提供任务信息（不同执行方式所需资源及预估时间）；时钟记录任务到达、排队、分配与完成时间；异构资源模块提供各集群可用资源、网络环境、已完成任务数量等信息。

调度器观测当前环境状态后，首先通过DeepSets模型提取关键特征并整合为特征向量，作为RL代理输入。智能体根据特征向量和当前策略决定是否立即分配任务或等待。当资源满足需求时，调度器输出□Cluster, Node, T□；否则将任务放入缓存队列等待，并重新进行调度。奖励生成器根据动作对系统目标的影响给出奖励，反馈给智能体以更新策略。

需要说明的是，图4中“奖励生成器”输出的奖励信号仅在训练阶段用于计算优势函数与策略梯度，并反馈更新PPO网络参数；在前向调度决策过程中，智能体仅依据DeepSets提取的环境特征向量进行动作选择，奖励不参与此次正向推理。该设计符合强化学习“决策-反馈-更新”的迭代训练范式。

4.1 基础设计

4.1.1 目标定义

基于前述云边端协同异构资源系统架构，每个集群通常包含多种异构计算资源，这些资源在计算能力、体系结构和芯片模型上各不相同。集群中每个节点的特征向量N表示为一个多元组，包含该节点的各种资源属性。具体地，对于集群中的任意节点j，其特征向量N_j可以定义为：

$$N_j=\left[C_{kj},C_{mj},G_{kj},G_{mj},N_{kj},N_{mj}\ldots,Disk_j,Net_j\right],\quad(2)$$

其中，C_{kj}, G_{kj},…表示节点j上异构处理器/加速器

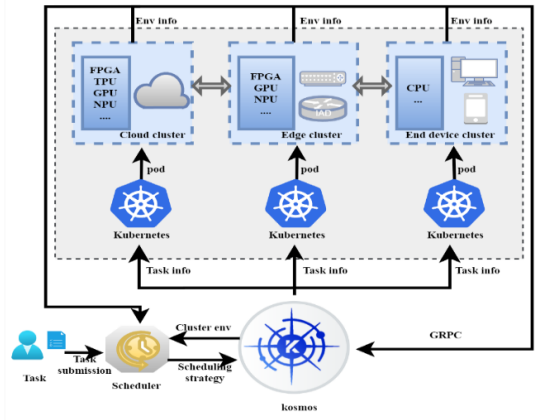


图3 异构资源云边端协同架构

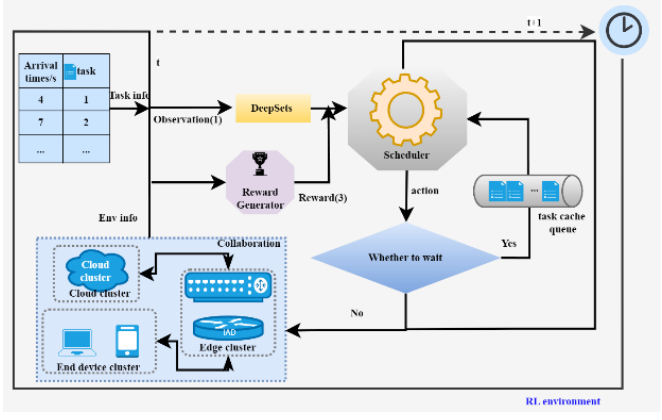


图4 DS-PPO模型调度策略概念图

(如CPU, GPU等)的可用核数。C_{mj}, G_{mj},…表示节点j上异构处理器/加速器(如CPU, GPU等)的可用内存，$Disk_j$表示节点j的可用存储空间，Net_j表示节点j的网络通信能力。为了更好地描述这些资源，本文将CNN模型推理任务集表示为T：

$$T=\langle T_1, T_2, T_3 \ldots T_k \rangle, \tag{3}$$

其中每个CNN模型推理任务i的需求可以表示为向量，表示为：

$$T_i=\begin{bmatrix} C_{rki1}, C_{rmi1}, G_{rki1}, G_{rmi1}, N_{rmi1}, \ldots, Disk_{ri1}, Runtime_{i1}, \ldots \\ C_{rkik}, C_{rmik}, G_{rkik}, G_{rmik}, N_{rkik}, N_{rmik}, \ldots, Disk_{rik}, Runtime_{ik} \end{bmatrix}, \tag{4}$$

其中，C_{rkix}, G_{rkix},…表示CNN模型推理任务i在第x种类型计算单元上运行所需的估计核数。C_{rmix}, G_{rmix}, …表示CNN模型推理任务i在第x种类型计算单元上运行所需的估计内存。$Disk_{rix}$表示CNN模型推理任务i在第x种执行策略下所需的估计存储空间。$Runtime_{ix}$表示CNN模型推理任务i在第x种执行策略下运行的估计持续时间。

假设云边端协同系统由K个集群组成，每个集群包含N_k个节点。决策变量$x_{t,n}$表示CNN模型推理任务t分配给节点n的情况。该变量为$0 \leqslant x_{t,n} < k$范围内的整数，其中k表示集群内存在异构计算单元的数量，每个值对应于节点上的特定执行策略。跨所有集群的CNN模型推理任务分配计划可以封装在以下矩阵表示中:

$$X=\left\{\begin{pmatrix} x_1^1 & \ldots & x_1^N \\ \vdots & \ldots & \vdots \\ x_T^1 & \ldots & x_T^N \end{pmatrix}\right\}. \tag{5}$$

其中上标$n=1,2,\ldots,N$表示节点编号，下标$t=1,2,\ldots,T$表示任务编号。矩阵元素x_t^n的值为整数，表示任务t在节点n上所使用的执行策略编号；若该任务未分配到节点n，则对应元素为0。

如果集群大小不同，则可以通过在矩阵中用零填充不可用芯片类型的位置来增强矩阵的通用性。将CNN模型推理任务t与节点n的分配表示为x_t^n，这表明本文调度算法的最终输出是确定当前任务的适当集群和节点，以及指定的执行策略s。

让N_i和T_i分别表示节点和CNN模型推理任务向量元组的第i个元素。分配后更新的向量可以表示为N_i'和T_i',更新规则定义如下：

$$N_i'=N_i-Net_i, \tag{6}$$

$$T_i'=T_i-Runtime_i. \tag{7}$$

约束条件：对于每个CNN模型推理任务i，更新任务嵌入（向量）必须小于/等于更新节点嵌入（向量）：

$$T_i' \leq N_i'. \tag{8}$$

对于平均等待和完成时间，本文保留了所有CNN模型推理任务的总和，并用显式索引来表示总和是针对单个CNN模型推理任务的：

$$Avg_{waiting\_time}=\frac{1}{m}\sum_{i=1}^{m} T_{waiting\_time,i}, \tag{9}$$

$$Avg_{completion\_time}=\frac{1}{m}\sum_{i=1}^{m}\left(T_{Runtime,i}+T_{waiting\_time,i}\right). \tag{10}$$

资源利用率u_j表示节点N_j上已分配资源的比例：

$$u_j=\frac{\sum_{j\in Nodes}\sum_{i\in Tasks} Resources\_Used_{ij}}{\sum_{j\in Nodes} Resources\_Available_j}. \tag{11}$$

其中$Resources_Used_{ij}$表示CNN模型推理任务i在节点j上执行时占用的资源量，$Resources_Available_j$表示节点j上可用的资源总量。

寻求最小化平均完成时间和平均等待时间，同时最大化资源利用率：

$$\min\left(\alpha\cdot Avg_{completion\_time}+\beta\cdot Avg_{waiting\_time}-\sigma\cdot u_j\right), \tag{12}$$

其中，α、β和σ为权重参数，满足$\alpha+\beta+\sigma=1$且在[0,1]范围内，用于平衡目标函数中的完成时间、等待时间和资源利用率。

在本文中使用平均等待时间、完成时间和资

源利用率作为核心评估指标，衡量用户体验以及系统性能。

4.1.2　基本设计

在本小节中，本文将着重介绍RL调度器的关键元素。

环境信息：表示系统中所有集群所有集点的集合，即 $Env_{info}=\{N_1,N_2,\ldots,N_k\}$，$k$ 表示所有节点的个数。

t：表示系统的内置时间步长，它记录了自系统开始调度首个CNN模型推理任务以来所经过的时间步长。当系统中没有待调度的CNN模型推理任务，或所有计算资源均处于空闲状态时，t 值会被重置为零，直至下一个CNN模型推理任务到达并重新启动计时。

任务信息：表示系统正在调度的CNN模型推理任务，记为 $Task_{info}$，当前等待调度的CNN模型推理任务及其等待时间可以表示为 $Task_{info}=\{T_1',T_2',\ldots,T_x'\}$，其中 x 表示CNN模型推理任务数，$T_i'=T_i+P,P$ 为CNN模型推理任务的等待时间。

动作空间：主要包括两种动作类型，第一种为任务分配动作，调度器直接输出在指定集群和节点上进行CNN模型推理任务的分配和执行，用< Cluster, Node, T>表示。第二种为等待动作，调度器允许当前任务等待一段时间 P 后再分配，以满足特定的调度策略或等待更好资源条件，用< Cluster, Node, T, P >表示。

奖励机制：奖励主要有两种类型，一种是-1，表示调度器当前的动作并未对环境造成任何实质性的改变，即当前的CNN模型推理任务被调度器安排为等待状态，而非立即执行。第二种类型则是大于0的正数值，其大小由调度器对优化目标所作决策的质量决定，并进一步影响后续的奖励值以及调度策略的调整。

强化学习过程如图5所示，对于给定的CNN

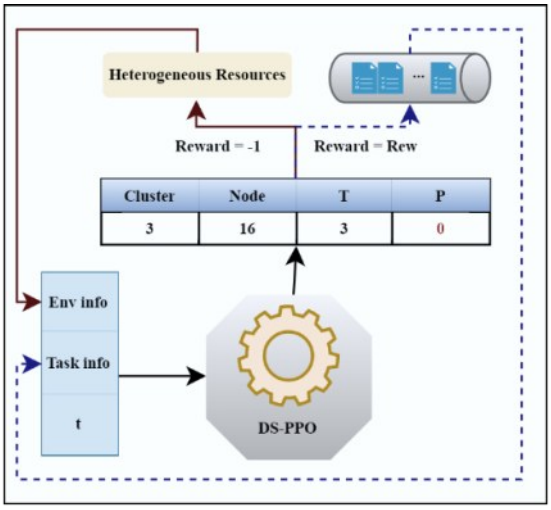


图5　基于RL的调度器内部工作

模型推理任务，每个动作的奖励被形式化为：

$$\mathrm{Re}\,ward(t)=\begin{cases}-1 & if\_wait\\ R(t) & if\_distribute\end{cases}. \tag{13}$$

$R_{(t)}$ 定义为：

$$G=\alpha\left(\frac{\sum T_{Runtime}}{m}+T_{Runtime}(t)\right), \tag{14}$$

$$H=\beta\left(\frac{\sum T_P}{m}+T_P(t)\right), \tag{15}$$

$$U\_factor=\sigma u_j(t), \tag{16}$$

$$R(t)=m+\frac{1}{G+H}+U\_factor, \tag{17}$$

其中，U_factor 表示资源利用率对奖励的贡献，$\sum T_{Runtime}$ 和 $T_{Runtime}(t)$ 分别代表截止当前时间完成的所有任务的总运行时间和任务 t 的预估运行时间。类似地，$\sum T_P$ 和 $T_P(t)$ 分别表示所有已完成任务的总等待时间和任务 t 的预估等待时间。变量 m 表示已完成的任务数量。

CNN模型推理任务调度优化问题本质上是一个复杂的混合整数线性规划（MILP）问题，具有非凸性和NP困难性，这导致在计算上寻找最优解变得极具挑战性[19]。为了有效应对这些挑战，启发式算法和强化学习被提出作为可行的解决方案。然而，在将这些方法应用于多维且复杂的CNN模型推理任务调度问题时，需要进行策

略优化，以确保能够适应问题的特定约束和条件，并找到接近最优的解。

4.2　策略优化

在云边端协同系统中，调度程序通过与环境（集群）的交互来学习如何做出最优决策。在每次交互中，智能体根据当前状态选择一个动作（如将某个CNN模型推理任务分配给某个节点），然后环境会给出相应的奖励或惩罚（反馈），智能体根据这些反馈来更新其决策策略[20]。DeepSets作为一种专门处理集合数据的深度学习模型，它的核心思想是通过两个函数ϕ和ρ分别实现特征提取和特征聚合，从而使网络对输入集合的顺序不敏感[21]。其具体数学定义如下：

$$DeepSets(X)=\rho\left(\sum x\in X\phi(x)\right), \quad (18)$$

其中，$X=\{x_1,x_2,...x_n\}$表示输入的集合，$\phi(x)$为特征提取函数，用于将每个元素x映射到高维特征空间，$\rho\left(\sum x\in X\phi(x)\right)$表示对所有特征进行聚合，并通过转换函数$\rho$输出最终的特征表示。DeepSets具有置换不变性、灵活性、高效性以及较强的泛化能力等特点。而PPO作为一种改进的策略梯度算法，通过引入概率比率剪切函数来限制策略更新的步长，从而提高训练的稳定性和效率[22]。通过将DeepSets与PPO算法相结合，可以利用DeepSets对集合数据的强大建模能力，处理云边端协同异构场景下的复杂CNN模型推理任务集合，进一步提升PPO算法在资源分配和任务调度中的性能，同时也提高了其泛化能力。

本文选用DeepSets作为特征提取模块，而非基于自注意力机制的Transformer结构，主要基于以下考量：云边端异构资源集合本质上具有无序性与动态可变性，DeepSets的置换不变聚合特性更贴合此类数据结构；同时，其线性计算复杂度相较于Transformer的二次复杂度，更适用于高并发、实时性要求高的调度场景。此外，DeepSets能够在不依赖任务-资源显式图结构或位置编码的情况下，有效提取集合整体表征，降低了模型构建的复杂性，有利于策略在未见环境中的泛化。未来工作中，我们将进一步系统比较不同特征提取方法（包括Transformer、图神经网络等）在云边端调度任务中的性能差异。

为适配云边端异构计算资源（如CPU、GPU、NPU等）的多样化特征，本文设计的DeepSets特征提取模块采用结构化多维特征表示，以刻画不同计算单元在算力、内存、存储与网络等方面的差异。每个节点n_i的特征向量$\mathbf{f}_i$涵盖计算能力（包括各类处理器/加速器的可用核心数、算力峰值）、内存容量与带宽、存储类型与容量、网络通信能力以及能耗估计等维度；相应地，每个CNN推理任务t_j的需求向量$\mathbf{d}_j$也按相同维度表达其资源需求与预估执行时间。为避免特征冗余与量纲不一致对训练稳定性的影响，本文在特征输入前进行了相关性筛选与动态归一化处理，仅保留与任务完成时间显著相关的特征，并对所有数值特征进行Z-score标准化。DeepSets通过其置换不变性天然支持异构资源集合的输入，每个计算单元作为集合中的一个元素，经特征提取函数ϕ映射为高维嵌入，再通过聚合函数ρ汇总为整体环境表征。此外，聚合过程中引入了轻量级注意力机制，以增强对当前任务最具亲和力的资源（如空闲GPU/NPU）的识别能力，从而使调度决策能够更精准地匹配任务需求与资源特性，提升系统整体效率与资源利用率。

尽管PPO算法在单任务、单环境训练中表现良好，但其策略网络在面对云边端系统中不同集群架构、动态任务类型及资源分布变化时，往往出现泛化能力不足的问题。为提升策略在跨集群、跨任务场景下的迁移能力，本文在结合DeepSets特征提取的基础上，引入环境感知的策略自适应机制与分层策略蒸馏方法。首先，通过在状态表征中显式编码集群拓扑、设备类型比例、网络延迟矩阵等元特征，使智能体能够感知

当前所处环境的结构性差异，并据此调整资源分配偏好。其次，采用分层策略结构：底层策略由DeepSets提取的通用资源特征驱动，学习跨异构资源的通用调度模式；上层策略则接收任务类型、集群负载等级等高层语义信息，实现对不同任务类型（如AI密集型、I/O密集型）的专用决策适配。训练过程中，引入多环境课程学习，依次在云集群、边缘集群及混合负载场景下进行渐进式训练，并在每次迭代中通过策略蒸馏将各环境下学得的策略知识融合至统一模型中，从而增强模型对未见集群与任务组合的快速适应能力。理论分析表明，本文所提出的环境感知策略自适应机制与分层策略蒸馏方法，有望显著提升DS-PPO在跨集群、跨任务类型场景下的策略迁移能力。未来工作中，我们计划通过从云侧集群向边缘集群迁移的实验验证该机制的有效性，预期DS-PPO在面对集群结构变化时，其性能下降幅度将显著低于传统PPO基线。

为了更好的了解策略优化过程，在表1中对重要符号进行说明。

其具体流程为：调度器接收到环境状态S_t，其中S_t为集合，包含了多个元素和特征（如集群中各节点异构计算资源状态、CNN模型推理任务信息等），表示为$S_t=\{s_{t,1},s_{t,2},...s_{t,n}\}$，对每个输入元素$s_{t,i}$应用特征提取函数：$\phi_{t,i}=\phi(s_{t,i})$，对所有元素特征进行求和聚合，然后通过聚合函数$\rho$转换为最终特征表示如下：

$$z_t=\rho\left(\sum_{i=1}^{n}\phi(s_{t,i})\right). \tag{19}$$

将特征向量z_t作为策略网络$z_t=\sum_{i=1}^{n}\phi_{t,i}$和价值网络的输入，得到策略网络输出动作的概率分布$\pi_\theta(a_t|z_t)$，价值网络估计当前状态值$V_\theta(z_t)$，智能体（调度器）在与环境交互时，生成并收集状态$z_t$、动作$a_t$、奖励$r_t$和下一状态$z_{t+1}$，并将这些数据存储在经验池中用于后续的训练，利用收

表1　策略优化重点参数介绍

Symbol	Descriptions
S	环境状态
$\phi(x)$	DeepSets模型中特征提取函数，用于将元素x映射到高维特征空间
$\rho(x)$	DeepSets模型中聚合函数，用于对所有特征聚合
z_t	DeepSets模型中通过聚合后，特征最终表现形式
$\pi_\theta(a_t\|z_t)$	策略网络
$V_\theta(z_t)$	表示价值网络对特征向量的估值
$V_\theta(z_{t+1})$	表示价值网络对下一个特征向量的估值
a_t、r_t	在t时刻的执行动作和奖励
A_t	优势函数
γ	优势函数中折扣因子（取值0~1），用于衰减未来奖励的重要性
$L^{CLIP}(\theta)$	分别为裁剪的策略损失当前价值网络参数
$L_V(\theta)$	价值函数的回归损失
$L_S(\theta)$	策略的熵损失
$L(\theta)$	最终损失函数
θ	当前价值网络参数
$r_t(\theta)$	新旧策略概率比
$\pi_\theta(a_t\|\phi(s_t))$	当前策略在当前状态下选取动作的概率
$\pi_{\theta\_old}(a_t\|\phi(s_t))$	旧策略在当前状态下选取相同动作的概率
ε、c_1、c_2	分别为限制策略更新幅度的小常数、调整价值函数的回归损失、策略的熵损失权重的超参数

集到的数据，通过价值网络计算优势函数为：

$$A_t = r_t + \gamma V_\theta(z_{t+1}) - V_\theta(z_t), \quad (20)$$

其中 r_t 为即时奖励，γ 为折扣因子，通过优势函数估计 A_t 可得PPO的损失函数：

$$L^{CLIP}(\theta) = E_t\Big[\min\big(r_t(\theta)A_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_t\big)\Big], \quad (21)$$

其中，$r_t(\theta)$ 为新旧策略概率比即：

$$r_t(\theta) = \frac{\pi_\theta(a_t|\phi(s_t))}{\pi_{\theta\_old}(a_t|\phi(s_t))}. \quad (22)$$

ε 为限制策略更新幅度的小常数，clip操作将 $r_t(\theta)$ 限制在[1-ε,1+ε]之间，同时加入价值函数的回归损失 L_V 和策略的熵损失 L_S 来增强探索。公式分别如下所示：

$$L_V(\theta) = E_t\Big[(V_\theta(z_t) - V_t^{target})^2\Big]. \quad (23)$$

得到最终综合损失函数为：

$$L(\theta) = L^{CLIP}(\theta) - c_1 L_V(\theta) + c_2 L_S(\theta), \quad (24)$$

其中 c_1 和 c_2 为调整各项权重的超参数。通过梯度下降法对 $L(\theta)$ 中网络参数 θ 进行更新，以最小化损失函数，DS-PPO调度策略在处理复杂输入数据、提高训练稳定性和策略性能上有显著优势，整个算法如表2所示。

5　实验

5.1　实验装置

实验基于移动云现网测试域的一个真实测试平台构建设计的云边端协同异构硬件资源调度系统。系统涉及的云、边缘和终端设备的硬件配置如下：

云侧集群:15台服务器，每台配备有8核CPU (Intel Xeon E5-2620 v4@2.10GHz)；64 GB RAM和4个Nvidia A100 GPU（16G）；10TB磁盘容量(SSD)。

边缘集群: 10台服务器，每台配备有2核CPU(A55 Arm core@1.6GHz); 8GB RAM和2个NVIDIA Jetson Xavier NX（8G）；1TB磁盘容量(HDD)。

终端设备集群: 8台终端服务器，每台配备有4核CPU(ARM cortex-A57@1.43GHz) ；16GB RAM和2个128-core Maxwell GPU @ 921MHz（4G）的NVIDIA Jetson Nano；500GB磁盘容量(SSD)。

通过Docker对云服务器、NVIDIA Jetson Xavier NX、NVIDIA Jetson Nano设备构建镜像，包含AI芯片（GPU、NPU）所需驱动程序和库函数，从而能够根据不同的AI芯片定制调度和执行策略来操作任务。

为全面验证DS-PPO调度系统在云边端异构环境中的泛化能力与鲁棒性，本文在实验设计中纳入了多种类型的计算任务（如表3所示）。尽管研究聚焦于高并发CNN模型推理任务，但实际CNN推理工作负载在执行过程中可能呈现出不同的资源消耗特征：例如，模型加载阶段表现为I/O密集型，特征提取阶段为计算密集型（AI密集型），而中间数据交换可能涉及内存与网络密集型行为。因此，通过构建涵盖AI密集型、内存密集型、I/O密集型、网络密集型及存储密集型的综合任务集，可更真实地模拟复杂CNN推理管线对异构资源的多元化需求，从而系统评估调度器在混合负载下的适应性与性能表现。

实验中为了更好地模拟CNN模型推理任务在云边端协同调度的实际场景，选择了几个常见的计算任务, 每种任务的任务信息如表3所示。

为了更好模拟高并发CNN模型推理任务分配与稳定CNN模型推理任务分配场景，引入两种任务队列生成方式：

第一种为标准CNN模型推理任务队列生成模式，在初始1000秒内，有100个任务随机到达，称“CNN模型推理稳定任务队列”，如移动云神眼项目中摄像头例行检查拍摄照片；第二种为CNN模型推理高并发任务队列生成模式，在初始1000秒时间间隔内，有2000个任务随机到

表2　DS-PPO调度策略算法

Algorithm 1: Training Algorithm

```
Input: Agent with DeepSets feature extractor and PPO algorithm, num_episodes, x tasks, max_time
for episode in range(num_episodes):
Initialize a new environment 'env' with x tasks, max_time as max_time, and time set to 0
Initialize total_reward to 0
Initialize PPO storage buffers
while env.time <= env.max_time:
batch_states = []
batch_actions = []
batch_log_probs = []
batch_rewards = []
batch_dones = []
for task in env.tasks:
if task.arrival_time == env.time:
# Extract features using DeepSets feature extractor
state = Agent.feature_extractor(env.tasks)
# Select action using PPO policy
action, log_prob = Agent.select_action(state)
# Execute action: get server based on action
server = action_to_server(action)
# Perform environment step: allocate task to selected server
next_state, reward, done = env_step(task, server)
if reward != -1:
# Store data for PPO optimization
batch_states.append(state)
batch_actions.append(action)
batch_log_probs.append(log_prob)
batch_rewards.append(reward)
batch_dones.append(done)
# Update total reward
total_reward += reward
# PPO optimization step using the collected batch
Agent.optimize_PPO(batch_states, batch_actions, batch_log_probs, batch_rewards, batch_dones)
# Increment the time step of the environment
env.time += 1
# Update epsilon if used (for exploration purposes)
if Agent.epsilon > Agent.epsilon_min:
Agent.epsilon *= Agent.epsilon_decay
# Check if all tasks are done
if all(task.done for task in env.tasks):
break
# Log episodic results if needed
log_results(episode, total_reward, env)
```

达，称为“CNN模型推理高并发任务队列”，如移动云智算项目中ResNet模型对CIFAR-10数据集10000张图像高频推理的计算任务。

实验中调度算法参数设置如表4所示：

在调度算法训练过程中，通过早停技术防止调度算法的训练模型过拟合，节省训练时间，如果最大奖励在连续70次没有增加，则终止训练。同时，Kubernetes内置调度算法仅支持对单集群内计算资源和任务进行匹配，不适用在云边端协同场景下进行比较，因此实验中对三类算法进行了性能及资源利用率的比较，对比的算法分别为：

（1）深度强化学习算法：一种基于深度强化学习的算法[22]，该算法和奖励函数设计主要针对服务器的磨损和同构计算资源和任务，设置的奖励函数与本文研究一致。

表3　AI密集型、内存密集型、I/O密集型任务、网络密集型任务、存储密集型任务的资源消耗指标

任务类型	设备	型号	C_{rk}	C_{rk}/MB	G_{rk}	G_{rm}/MB	N_{rk}	N_{rm}/MB	Disk/MB	Runtime/s
AI密集型	云服务器集群	CPU	0.53	1664	0.00	0.00	0.00	0.00	25	65
		GPU	0.21	1021	0.43	1898.00	0.00	0.00	8	12
	边缘设备集群	CPU	1.21	768	0.00	1898.00	0.00	0.00	28	1021
		NPU	0.45	1024	0.00	0.00	0.48	1024.00	10	101
	终端设备集群	CPU	1.11	888	0.00	0.00	0.00	0.00	34	892
内存密集型	云服务器集群	CPU	0.12	2212	0.00	0.00	0.00	0.00	23	11
	边缘设备集群	CPU	0.18	2021	0.00	0.00	0.00	0.00	18	42
	终端设备集群	CPU	0.81	1989	0.00	0.00	0.00	0.00	19	28
I/O密集型	云服务器集群	CPU	2.61	1232	0.00	0.00	1.34	124.00	99	234
		GPU	1.58	893	0.98	937.00	0.67	79.00	83	125
	边缘设备集群	CPU	1.23	467	0.00	0.00	2.31	576.00	132	89
		NPU	2.05	1032	1.14	683.00	1.87	114.00	257	134
	终端设备集群	GPU	3.56	779	1.74	1145.00	0.95	436.00	65	76
网络密集型	云服务器集群	CPU	1.68	345	0.00	0.00	3.41	1158.00	23	101
	边缘设备集群	CPU	2.67	1156	0.00	0.00	2.65	2512.00	78	231
		GPU	1.99	886	1.54	876.00	2.21	1028.00	144	122
	终端设备集群	CPU	0.65	2410	0.00	0.00	1.46	1968.00	86	198
存储密集型	云服务器集群	CPU	0.09	221	0.00	0.00	0.00	0.00	2098	18
	边缘设备集群	CPU	0.23	80	0.00	0.00	0.00	0.00	1989	125
	终端设备集群	CPU	0.98	101	0.00	0.00	0.00	0.00	2012	56

表4　强化学习参数

算法参数	参数值
Gamma 折扣因子γ	0.990
Learning_Rate 学习率(LR)	0.001
N_Steps 采集时间步数	2408.000
Batch Size 样本容量大小	64.000
N_Epochs 训练轮次	10.000
Clip_Range 策略幅度	0.200
Ent_Coef 熵系数	0.010

（2）启发式算法：一种混合优化算法[21][23]，通过支持向量机对服务器进行分配，通过粒子群算法寻找最优服务器，并通过使用GWO算法寻找运行时间最短的调度策略。另一种算法是一种使用S型函数进行惯性权重优化的粒子群算法[24]。

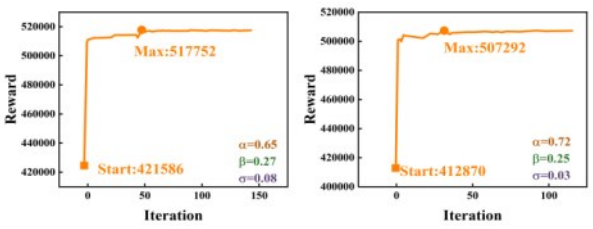
（3）贪婪算法：首次适用算法（FF），是一种经典调度算法，采取贪婪逻辑，按先来先服务执行顺序，考虑在任务到达时获取最大奖励[25]。

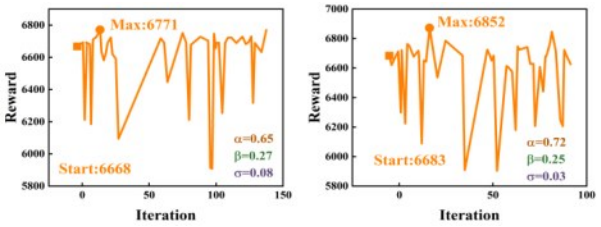
本研究实验聚焦于任务均匀到达的稳定与高并发场景，以验证调度算法在持续负载下的性能。未来工作将涵盖突发任务到达等更极端的负载模式，以进一步测试系统的鲁棒性与弹性。

5.2　DS-PPO模型参数与收敛性分析

实验采用层次搜索法对DS-PPO模型目标函数中α、β、σ进行设置，分别训练CNN模型高并发任务队列和CNN模型推理稳定队列的DS-PPO模型，观察α、β、σ对模型训练效果的影响，如图6所示：

(a)CNN模型推理高并发任务队列下，α对模型训练效果的影响 (b)CNN模型推理稳定任务队





列下，α对模型训练效果的影响

其中α和β的取值主要影响模型在性能上更加侧重于任务平均完成时间还是侧重于任务平均等待时间，σ的取值是模型对于异构计算资源利用率的反馈。目标函数中权重系数α、β、σ的选取本质上反映了系统对不同优化目标的偏好权衡。本文采用层次分析法结合实验网格搜索确定其取值，具体依据如下：

（1）业务优先级导向：在云边端协同推理场景中，任务完成时间直接影响用户体验，故赋予较高权重（α较大）；资源利用率影响运营成本，赋予中等权重（σ居中）；等待时间在可接受范围内可适当放宽，故权重较低（β较小）。

（2）量纲归一化处理：为避免不同指标量纲差异影响权重有效性，实验前已对完成时间、等待时间、资源利用率三项指标进行Z-score标准化，使其具有可比性。

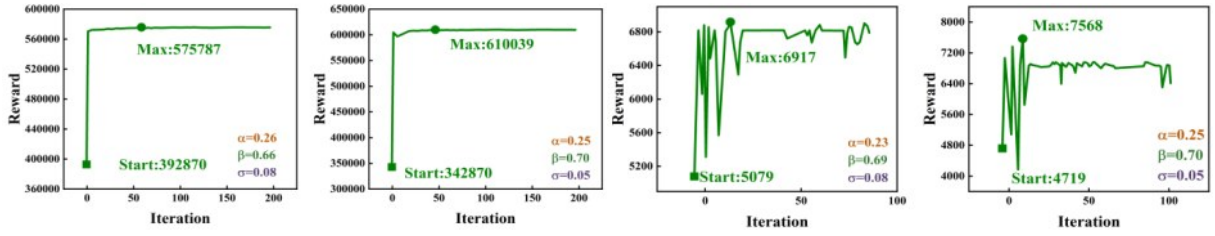
（3）网格搜索验证：在预实验阶段，我们在$\alpha+\beta+\sigma=1$的约束下，以0.1为步长遍历所有权重组合，选取在验证集上能使综合目标函数收敛最快、最终奖励最高的权重组合作为初始值。

从图6中可以看出几种现象：

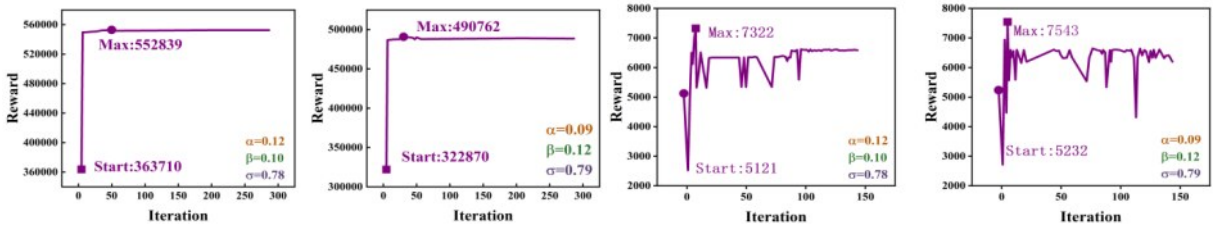
现象一：无论性能还是资源利用率，DS-PPO在高并发队列中的优化效果明显优于稳定队列。

现象二：性能上，模型在优化队列中任务的等待时间方面优于优化队列中任务的完成时间。

从图6中可以明显看出，每组数据的收敛曲线第二行（CNN模型推理稳定任务队列）明显劣于第一行（CNN模型推理高并发任务队列）。主要是在对稳定任务队列进行多目标优化时，容易出现过拟合。稳定任务队列的任务发布在时间线上更加分散。从图6的数据可以看出，当DS-PPO关注完成时间时，高并发队列和稳定队列平均效率分别提高22.84%和2.03%；关注等待时间时，提升62.24%和48.28%；关注计算资源利用率时，提升51.99%和43.58%。



(c) CNN模型推理高并发任务队列下，β对模型训练效果的影响 (d)CNN模型推理稳定任务队列下，β对模型训练效果的影响



(e) CNN模型推理高并发任务队列下，σ对模型训练效果的影响 (f)CNN模型推理稳定任务队列下，σ对模型训练效果的影响

图6 CNN模型推理高并发任务队列和稳定任务队列在DS-PPO模型的训练过程和收敛性能评价

5.3　CNN 模型推理任务平均完成时间分析

本节通过任务平均完成时间评价 DS-PPO 的优化能力，并与三类基准算法对比。在高并发与稳定队列环境下的结果如图 7 所示。从图 7 中可以发现几个现象:

现象一：对于 CNN 模型推理稳定任务队列，强化学习训练方法优于启发式学习、贪婪学习训练方法。然而，当应对到 CNN 模型推理高并发任务队列时，强化学习表现出优异的性能。

现象二：在本文设计的云边端异构环境中，不同 RL 算法的效果差异相对有限。

对于现象一，由图 7 可得，在稳定队列中，Sigmoid-PSO 平均完成时间为 14.89 秒，优于所有算法平均值，S-P-GWO 也优于 RL 算法，这是因为任务分布分散、搜索空间较小、最优上界固定，且仅 100 个任务使得 RL 难以充分学习，容易过拟合，而启发式算法易于在小规模空间找到较优解。

在高并发队列中，2000 个任务提供了更高维搜索空间，使 RL 优势显现。DS-PPO 在关注完成时间时平均仅需 27.46 秒，为各算法中最优；启发式与贪婪算法平均完成时间为 103.23 秒，比 DS-PPO 差 275.93%。同时，启发式和贪婪算法在可移植性和在线调度能力方面也不如事先训练好的 RL 模型。

对于现象二，稳定队列下，DS-PPO 与 DRL-DQN、DRL-REINFORCE 差异不大；在高并发队列下差异最大仅 14.16 秒。该差异是多次训练中选取最优模型的结果，体现了 DS-PPO 设计在稳定性、效果优势。

5.4　CNN 模型推理任务等待完成时间分析

本节从平均等待时间角度评估 DS-PPO 的优化能力，对比结果如图 8 所示。由图 8 可以发现几个现象:

现象一：在稳定队列和高并发队列中，仅针对等待时间进行优化的 RL 算法均优于启发式和贪婪算法。

现象二：在两种队列中，FF 贪婪算法的平均等待时间均显著偏高。

原因在于，RL 算法通过与环境交互学习策略，能适应环境变化，仅通过改变优化目标就可减少平均等待时间。FF 算法则只关注当前任务完成时间的最优，不考虑全局收益和未来任务：即使有稍差的节点空闲，也倾向等待更优节点，这会导致随着任务积累等待时间不断增长，后续复杂任务因先到先得而分配给较差节点，造成负载不均和资源利用率低下。

5.5　计算资源利用率分析

最后，本节评估 DS-PPO 对计算资源利用率的优化能力，对比结果如图 9 所示。由图 9 中可以发现:

现象一：在稳定队列中，针对资源利用率进行学习的 RL 与启发式、贪婪算法差异不明显；而在高并发队列中，RL 在资源利用率上的优势明显。

现象二：在高并发队列环境中，DS-PPO 在资源利用率方面优于其他 RL 算法以及启发式与贪婪算法。

对于现象一，稳定任务队列中各 CNN 推理任务的资源需求较为固定，启发式和贪心算法可依据预定义规则快速完成较优分配，强化学习优势不明显。而在高并发任务队列中，资源需求波动大且变化频繁，启发式算法需要不断调整，易产生次优分配；强化学习则通过持续与环境交互、更新策略，可动态优化资源分配，更适合不确定、强动态场景。

对于现象二，在高并发场景下，2000 个任务相较 100 个任务形成维度更高、结构更复杂的搜索空间，更能体现强化学习的优势。以计算资源利用率为例，DS-PPO 模型相比其他强化学习算法以及启发式、贪心算法整体利用率提升超过 38.86%，表现最优。DS-PPO 兼顾离散与连续动

作空间，可统一处理两类决策：

(1) 离散分布空间：在任务分配过程中，有些决策是离散的（例如选择哪个服务器处理任务）。

(2) 连续分布空间：在资源分配过程中，有些决策是连续的（例如分配多少 CPU 资源、内存等）。

DS-PPO 模型能够提供更精细的资源分配策略，相比只处理单一分布空间的算法具有显著优势。本文采用 DeepSets 神经网络进行特征优化，DeepSets 能有效从任务与环境约束中提取关键信息，使 DS-PPO 能动态调整资源分配策略，在 CNN 模型推理高并发任务队列下仍保持较高的计算资源利用率。同时，DeepSets 提取的特征有助于模型快速识别资源需求变化，从而及时调整分配策略并进一步优化资源利用率。

6 结束语

本文提出一种云边端协同计算框架，充分利用跨云、边、端的异构算力：形式化异构资源任务调度问题，基于扩展的 Kubernetes 与 Kosmos 设计协同任务框架，并提出基于强化学习泛化能力优化的 DS-PPO 算法作为核心调度器。实验表明，在多种 CNN 推理任务设置下，调度器可以有效学习以最大化奖励，尤其在高并发任务队列下显著缩短任务完成时间并提升资源利用率。该方法可扩展到服务器成本、能耗等多目标优化。如何在多目标间平衡优先级以及在数学模型与超参数设计上进一步改进，是未来工作重要方向。此外，尽管本文未直接与基于注意力机制的强化学习调度算法（如 Trans-Sched）进行实验对比，但从方法学角度分析，DS-PPO 与这类方法的核心差异在于：DS-PPO 通过 DeepSets 实现置换不变的特征聚合，更适合动态资源集合的调度场景；而注意力机制虽能建模任务-资源间的复杂依赖，但计算开销随规模增长较快。未来工作中，我们将进一步设计实验，系统比较不同表征学习机制在云边端异构调度任务中的性能与泛化表现。

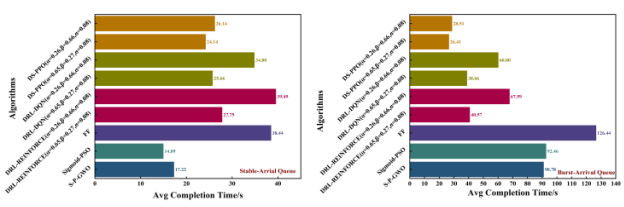


图7 不同队列环境下对比实验：平均完成时间

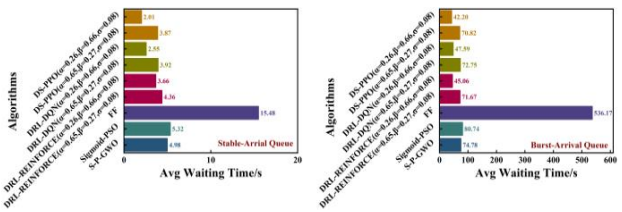


图8 不同队列环境下对比实验：平均等待时间

(a)推理稳定任务队列 (b)推理高并发任务队列 (a)推理稳定任务队列 (b)推理高并发任务队列

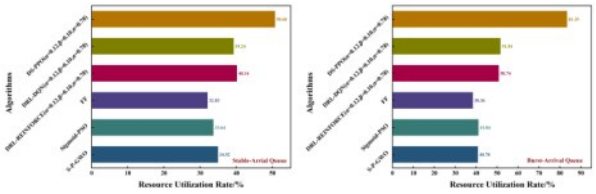


图9　不同队列环境下对比实验：计算资源利用率
(a)CNN 模型推理稳定任务队列 (b)CNN 模型推理高并发任务队列

参考文献：

[1] Hosseinzadeh M, Azhir E, Lansky J, et al. Task scheduling mechanisms for fog computing: a systematic survey[J]. IEEE Access, 2023, 11: 50994-51017.

[2] Jiang M, Wu T, Wang Z, et al. A multi-intersection vehicular cooperative control based on end-edge-cloud computing[J]. IEEE Transactions on Vehicular Technology, 2022, 71(3): 2459-2471.

[3] Ren J, Jiang H, Shen X, et al. Editorial of ccf transactions on networking: special issue on intelligence-enabled end-edge-cloud orchestrated computing[J]. CCF Transactions on Networking, 2020, 3(3): 155-157.

[4] Ren J, Zhang D, He S, et al. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet[J]. ACM Computing Surveys (CSUR), 2019, 52(6): 1-36.

[5] Attiya I, Abd Elaziz M, Abualigah L, et al. An improved hybrid swarm intelligence for scheduling IoT application tasks in the cloud[J]. IEEE Transactions on Industrial Informatics, 2022, 18 (9): 6264-6272.

[6] Alsaidy S A, Abbood A D, Sahib M A. Heuristic initialization of PSO task scheduling algorithm in cloud computing[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(6): 2370-2382.

[7] Wang L, Pan Z, Wang J. A review of reinforcement learning based intelligent optimization for manufacturing scheduling[J]. Complex System Modeling and Simulation, 2021, 1(4): 257-270.

[8] Houssein E H, Gad A G, Wazery Y M, et al. Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends[J]. Swarm and Evolutionary Computation, 2021, 62: 100841.

[9] Yuan H, Bi J, Zhou M C. Multiqueue scheduling of heterogeneous tasks with bounded response time in hybrid green IaaS clouds[J]. IEEE Transactions on Industrial Informatics, 2019, 15(10): 5404-5412.

[10] Zhou J, Sun J, Cong P, et al. Security-critical energy-aware task scheduling for heterogeneous real-time MPSoCs in IoT[J]. IEEE Transactions on Services Computing, 2019, 13(4): 745-758.

[11] Carrión C. Kubernetes scheduling: Taxonomy, ongoing issues and challenges[J]. ACM Computing Surveys, 2022, 55(7): 1-37.

[12] Narayanan D, Santhanam K, Kazhamiaka F, et al. {Heterogeneity-Aware} cluster scheduling policies for deep learning workloads[C]//14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). 2020: 481-498.

[13] Feng J, Zhang W, Pei Q, et al. Heterogeneous computation and resource allocation for wireless powered federated edge learning systems[J]. IEEE Transactions on Communications, 2022, 70(5): 3220-3233.

[14] Zhong Z, Buyya R. A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources[J]. ACM Transactions on Internet Technology (TOIT), 2020, 20(2): 1-24.

[15] Abdulazeez D H, Askar S K. Offloading mechanisms based on reinforcement learning and deep learning algorithms in the fog computing environment[J]. Ieee Access, 2023, 11: 12555-12586.

[16] Jung Y, Park J. Bayesian Convolutional Deep Sets with Task-Dependent Stationary Prior[C]//International Conference on Artificial Intelligence and Statistics. PMLR, 2023: 3795-3824.

[17] He H, Meng X, Wang Y, et al. Deep reinforcement learning based energy management strategies for electrified vehicles: Recent advances and perspectives[J]. Renewable and Sustainable Energy Reviews, 2024, 192: 114248.

[18] Xie B, Bian Y, Chen Y, et al. Enhancing neural subset selection: Integrating background information into set representations[J]. arXiv preprint arXiv:2402.03139, 2024.

[19] Kim C H, Kim S J, Choi H L. Heterogeneous satellite task scheduling with revisit time minimization using milp formulation[C]//AIAA SCITECH 2024 Forum. 2024: 0747.

[20] Cui H, Tang Z, Lou J, et al. Latency-aware container scheduling in edge cluster upgrades: A deep reinforcement learning approach[J]. IEEE Transactions on Services Computing, 2024, 17 (5): 2530-2543.

[21] Cheng Y, Guo Q, Wang X. Proximal policy optimization with advantage reuse competition[J]. IEEE Transactions on Artificial Intelligence, 2024, 5(8): 3915-3925.

[22] Du Y, Li J. A deep reinforcement learning based algorithm for a distributed precast concrete production scheduling[J]. International Journal of Production Economics, 2024, 268: 109102.

[23] Yassami M, Ashtari P. A novel hybrid optimization algorithm:

Dynamic hybrid optimization algorithm[J]. Multimedia Tools and Applications, 2023, 82(21): 31947-31979.

[24] Premkumar M, Sowmya R, Ramakrishnan C, et al. An efficient and reliable scheduling algorithm for unit commitment scheme in microgrid systems using enhanced mixed integer particle swarm optimizer considering uncertainties[J]. Energy Reports, 2023, 9: 1029-1053.

[25] Rouskas G N. First-Fit: A universal algorithm for spectrum assignment[C]//GLOBECOM 2023-2023 IEEE Global Communications Conference. IEEE, 2023: 2123-2128.

[作者简介]

张阔（1988-），男，香港理工大学机械工程硕士，现任中国铁塔股份有限公司通信技术研究院高级经理、高级工程师，主要从事边缘计算、算力网络方向的技术研究和研发创新。

闫亚旗（1988-），男，北京理工大学电子信息硕士，现任中国铁塔股份有限公司通信技术研究院高级经理、高级工程师，主要研究方向为物联网、边缘计算、算力网络相关技术及产品创新。

安颖（1992-），女，北京邮电大学信息安全硕士，现任中国铁塔股份有限公司通信技术研究院经理，主要研究方向为算力网络、智算网络架构和云原生技术等。

董玉池（1988-），男，电子科技大学计算机硕士，现任中国铁塔股份有限公司通信技术研究院高级经理，主要研究方向为边缘计算、算力网络和在网计算等。

张民贵（1980-），男，清华大学计算机科学与技术博士，现任中国铁塔股份有限公司通信技术研究院技术总监、高级工程师，主要研究方向为算力网络、工业互联网、边缘计算、数据中心网络等。

袁刚（1989-），男，南京邮电大学通信与信息系统专业硕士，现任中移（苏州）软件技术有限公司产品经理，主要研究方向为云原生+智算技术体系构建、产品化落地。