基于 vGPU 性能干扰感知的大模型推理负载资源高效配置方法

张 虎[1,2,3],孙明辉[2,3],刘 杨[4],戴鸿君[1*],王继彬[2,3],张有利[2,3]

(1. 山东大学集成电路学院,山东济南 250101;
2. 齐鲁工业大学(山东省科学院)山东省计算中心(国家超级计算济南中心)算力互联网与信息安全教育部重点实验室,山东济南 250013;
3. 山东省算力互联网与服务计算重点实验室,山东省基础科学研究中心(计算机科学),山东济南 250103;4. 北京邮电大学,北京 100876)

摘 要: 人工智能(Artificial Intelligence,AI)技术的快速演进推动了开源大模型在多元化场景中的规模化应用. 然而,随着图形处理器(Graphics Processing Unit,GPU)单卡性能的提升,在支持中小规模大模型推理负载时,GPU 资源易出现闲置现象,导致整体算力利用率不足. 为提升数据中心 GPU 资源使用效率,业界普遍采用时空共享或虚拟 GPU(Virtual GPU,vGPU)技术实现算力复用,其中 vGPU 凭借细粒度资源划分与安全隔离特性,已成为数据中心向多租户、多任务提供 GPU 资源服务的主流方案. 然而,GPU 资源共享技术不可避免地引入任务负载之间的性能干扰,尤其是大模型推理负载所需资源具有动态性和突发性. 在不考虑性能干扰的情况下,会导致推理负载延迟显著增加,甚至引发服务质量目标(Service Level Objective,SLO)违约,影响大模型服务的稳定性与用户体验. 针对这一关键挑战,本文提出了一种基于 vGPU 性能干扰感知的大模型推理负载资源高效配置方法. 该方法首先通过大规模并发推理实验,构建了涵盖不同参数规模大模型、不同负载组合、不同负载强度下的多维性能表征数据集;在此基础上,建立了综合考虑推理模型特征、硬件支撑信息及系统监控指标的轻量化性能干扰预测模型,既保证了对关键性能指标的精准估计,也满足了资源配置决策的实时性需求. 基于该预测模型,本文进一步设计了基于约束优化的经济型资源配置算法,以最小化 GPU 资源分配量为目标函数,以推理延迟不超过 SLO 阈值、吞吐量满足业务需求为约束条件,通过动态调整各负载的 vGPU 资源分配比例,实现了在满足推理负载质量约束的前提下 GPU 资源分配优化. 实验部分构建了包含两类六种典型大模型的混合负载测试环境,并在 NVIDIA A100 和 RTX6000 硬件平台与 HAMi vGPU 方案上,与传统 GPU 配置策略进行了对比验证. 实验结果表明,所提方法在严格满足 SLO 约束的前提下,相较主流方案可降低超过 20% 的 GPU 资源成本开销,验证了其在大模型推理场景下的有效性与经济性,为数据中心提升 GPU 资源利用效率、降低人工智能服务部署成本、促进开源大模型的规模化普及应用提供了重要技术支撑.

关键词: 大模型;推理负载;vGPU;性能干扰;资源配置

基金项目: 国家重点研发计划(No.2024YFB2906605);山东省重点研发计划(No.2024CXGC010113);齐鲁工业大学(山东省科学院)科教产融合试点工程重大创新类项目(No.2024ZDZX08)

中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2025)11-3836-16

电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250468

Resource-Efficient Configuration Method for Large Model Inference Loads Based on vGPU Performance Interference Awareness

ZHANG Hu[1,2,3], SUN Ming-hui[2,3], LIU Yang[4], DAI Hong-jun[1*], WANG Ji-bin[2,3], ZHANG You-li[2,3]

(1. *School of Integrated Circuits, Shandong University, Jinan, Shandong* 250101, *China*;
2. *Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, Shandong* 250013, *China*;
3. *Shandong Provincial Key Laboratory of Computing Power Internet and Service Computing, Shandong Fundamental Research Center for Computer Science, Jinan, Shandong* 250103, *China*; 4. *Beijing University of Posts and Telecommunications, Beijing* 100876, *China*)

Abstract: The rapid evolution of artificial intelligence (AI) has propelled the large-scale application of open-source large language model across diverse scenarios. However, with the substantial performance boost of individual graphics pro-

收稿日期:2025-06-01;录用日期:2025-11-14;责任编辑:覃怀银
*通讯作者:戴鸿君

cessing unit (GPU), resources often suffer from idling when serving inference workloads for small- to medium-sized LLM, leading to insufficient overall computing utilization. To enhance GPU efficiency in data centers, spatial-temporal sharing or virtual GPU (vGPU) technologies are widely adopted for resource multiplexing. Notably, vGPU has emerged as the mainstream solution for providing GPU services to multi-tenant and multi-task environments, owing to its fine-grained resource partitioning and robust security isolation. Nevertheless, GPU resource sharing inevitably introduces performance interference among workloads, particularly given the dynamic and bursty resource demands characteristic of LLM inference. Neglecting such interference can lead to a significant surge in inference latency and trigger service level objective (SLO) violations, thereby compromising the stability and user experience of LLM services. To address this critical challenge, this paper proposes an efficient resource provisioning method for LLM inference workloads based on vGPU performance interference awareness. First, we construct a multi-dimensional performance characterization dataset through large-scale concurrent inference experiments, covering various LLM parameter sizes, workload co-location combinations, and intensities. On this basis, a lightweight performance interference prediction model is established, incorporating model features, hardware specifications, and system monitoring metrics. This model ensures precise estimation of key performance indicators while meeting the real-time requirements of resource decision-making. Leveraging this prediction model, we further design a constraint-optimization-based economic resource allocation algorithm. With the objective of minimizing GPU resource consumption and constraints ensuring inference latency remains within SLO thresholds and throughput meets business demands, the algorithm optimizes GPU resource allocation by dynamically adjusting the vGPU partition ratios for each workload. We evaluate the proposed method in a mixed workload environment comprising two categories and six typical LLMs. The experiments are conducted on NVIDIA A100 and RTX6000 platforms utilizing the HAMi vGPU solution, benchmarking against traditional GPU provisioning strategies. Experimental results demonstrate that the proposed method reduces GPU resource overhead by over 20% compared to mainstream schemes while strictly adhering to SLO constraints. These findings validate the effectiveness and economic viability of the approach in LLM inference scenarios, providing significant technical support for data centers to enhance GPU utilization, reduce AI service deployment costs, and facilitate the large-scale adoption of open-source LLM.

Key words: large models; inference workloads; vGPU; performance interference; resource allocation

Foundation Item(s): National Key Research and Development Program of China (No.2024YFB2906605); Key Research and Development Plan of Shandong Province (No.2024CXGC010113); Pilot Project for Integrated Innovation of Science, Education, and Industry of Qilu University of Technology (Shandong Academy of Sciences) (No.2024ZDZX08)

1　引言

近年来,以GPT[1]、DeepSeek[2]为代表的大语言模型在自然语言处理、多模态推理等领域取得突破性进展,推动着人工智能(Artificial Intelligence,AI)技术向规模化、服务化方向快速演进. 据统计,2024年全球AI服务器市场规模已达1 251亿美元,到2028年有望达到2 227亿美元[3]. 其中GPU(Graphics Processing Unit)作为大模型推理的核心算力载体[4],其资源利用率与服务质量水平成为影响云数据中心运营成本的关键因素. 然而,当前主流图形处理器(GPU)加速卡(如NVIDIA A100/H100)的单卡算力已突破300 TFLOPS,且配备显存也在不断增大. 在这种情况下,单卡GPU支持百亿参数及以下规模的模型推理时,普遍存在算力闲置的问题[5]. 这种资源冗余与日益增长的推理负载资源需求形成尖锐矛盾,促使业界积极探索GPU虚拟化技术以实现算力高效复用.

当前,业界对GPU资源共享方案进行了非常多的探索. 正如2.1节中提到的,在主流GPU资源共享方案中,基于硬件隔离的vGPU技术因其细粒度资源划分和强安全隔离特性,已成为阿里云、AWS等云服务商提供AI推理服务的主流基础设施. 然而,NVIDIA没有开源其虚拟GPU(Virtual GPU,vGPU)的实现代码,因此针对NVIDIA显卡的资源共享机制大多使用计算统一设备架构(Compute Unified Device Architecture,CUDA)请求劫持的方式,实现简单的时间切片或显存、算力切分,并且只实现了显存和算力能力的细粒度切分,而对底层的缓存、流处理器(Streaming Multiprocessor,SM)等资源并没有进行切分. 图1展示了一个典型的vGPU架构. 该框架中的节点代理和vGPU管理器对CUDA的驱动进行了重写,通过从容器内部对CUDA请求的劫持,直接在vGPU管理器中判断和管理资源使用情况,从而支持多个应用通过容器共享物理GPU资源并进行算力和显存的动态分配.

这种GPU资源共享机制会引发多维度的性能干扰问题. 首先,多实例间的共享缓存竞争导致内存访问延迟非线性增长;其次,计算单元调度冲突造成内核执行流水线阻塞;第三,动态电压频率调节机制引发的全局频率降级进一步放大性能衰减效应. 正如5.2节中性能

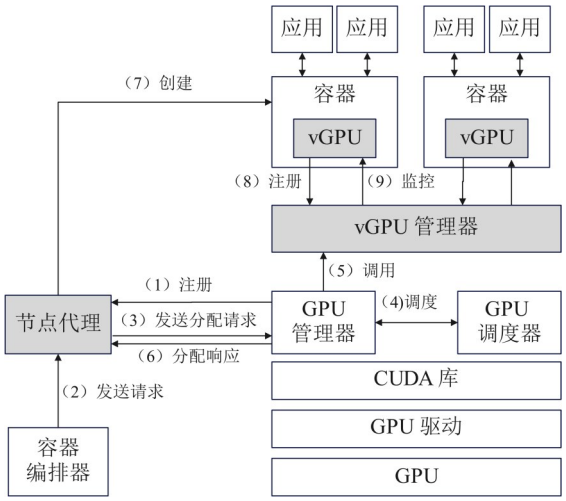


图1 vGPU的实现架构图

干扰实验中证明的那样，从独占GPU推理到4个大模型共享GPU时，关键指标延迟增长了10%~100%，严重影响服务质量. 这种性能干扰的隐蔽性，使得传统基于资源预留的静态配置方法难以在服务质量和资源效率间取得平衡.

现有研究在vGPU性能干扰治理方面已取得部分进展：Dhakal等人[6]提出了面向可扩展推理平台的GPU控制空间共享技术，通过控制流处理器分配比例实现计算资源隔离，但未考虑共享缓存层次的影响；Choi等人[7]提出了基于GPU时空共享的机器学习推理服务框架，采用线性回归模型预测机器学习模型推理任务场景的延迟增量，但其扩展性受制于组合爆炸问题；多实例GPU服务（Multi-Instance GPU Serving，MIG-Serving）[8]依托硬件分区机制实现物理隔离，却牺牲了资源按百分比灵活切分的能力. 这些方法普遍存在3方面局限：(1)干扰特征表征维度单一，缺乏对计算、显存、功耗等耦合因素的协同分析；(2)性能预测模型依赖大规模离线剖析，难以适应动态负载场景；(3)多是考虑深度神经网络（Deep Nueral Network，DNN）模型的推理场景，对大模型推理场景的特征分析及对应的资源分配策略研究较少.

本文提出基于vGPU性能干扰感知的大模型推理资源高效配置方法，主要贡献如下：(1)通过真实环境下在线推理实验，构建了多维度性能表征数据集；(2)系统地量化了硬件配置、推理负载特征以及实时监控数据对推理负载关键指标的数学关系，设计了轻量级vGPU性能干扰预测模型，实现秒级干扰估算；(3)在此基础上，建立带约束的优化目标，创新性地将资源配置问题转化为多目标优化任务，在严格满足服务质量目标（Service Level Objective，SLO）的前提下优化GPU资源占用. 实验表明，本方法在HAMi vGPU平台上支持混合负载部署时，较传统方案降低超20%的资源成本.

2 相关工作

2.1 GPU资源共享方案

早期在大模型推理资源管理中，多采用静态分配方式，即根据模型的最大资源需求预先分配固定的GPU资源. 然而，这种方式在面对不同参数规模的大模型以及推理负载动态变化时，存在明显的资源浪费问题. 例如，对于中等及以下参数规模的大模型，其在推理过程中可能并不需要始终占用整块GPU卡的资源，但传统方法仍会按照最大需求进行分配，导致大量资源闲置.

随着对资源利用效率的持续关注，一些研究开始尝试基于负载预测的动态资源分配策略. 通过对推理负载的历史数据进行分析，预测未来负载的变化趋势，进而动态调整资源分配. 但这些方法往往依赖于较为准确的负载预测模型，且在面对复杂的多模型混合推理场景时，预测精度和资源分配的实时性难以保证. 因此，学术界开始关注如何通过虚拟化技术提升GPU资源的利用效率，以解决负载预测方法的局限性. 虚拟化技术能够将一块物理GPU划分为多个虚拟GPU，实现细粒度的资源划分和复用，为解决GPU资源闲置问题提供了新的思路，并提供更灵活的资源管理方式和更高的资源利用率.

在GPU虚拟化与资源共享领域，研究者提出了多种解决方案. HAMi[9]是一个异构智算资源虚拟化中间件，能够实现在Kubernetes中统一管理、调度和共享多种异构计算设备，并提供资源隔离和高效利用能力. Shi等人[10]提出了在虚拟机内的GPU硬件加速框架（vCUDA），通过应用程序编程接口（Application Programming Interface，API）拦截和重定向以及专用的远程过程调用系统，实现了虚拟机中对GPU硬件的高效访问和资源共享，显著提升了虚拟化环境下的高性能计算能力. Lin等人[11]提出了GPU虚拟化框架（qCUDA），通过基于Virtio框架的半虚拟化技术，实现了高性能的GPU虚拟化，显著提高了虚拟机中CUDA程序的执行效率，支持Linux和Windows环境. 基于内核隔离vGPU的容器共享技术（cGPU）[12]通过基于内核的容器共享技术，实现了GPU资源的高效隔离与共享，显著提升了资源利用率，降低了成本，并支持多种业务场景. 腾讯虚拟化GPU服务（qGPU）[13]通过腾讯云容器服务的开源弹性GPU框架，实现了GPU资源的细粒度隔离与共享，支持混合部署，提高了资源利用率，降低了成本，并保持了高性能. Goswami等人[14]提出了公平共享中间件技术（GPUShare），通过软件机制实现GPU时间切片和资源让出，显著提升了多租户环境下GPU资源的公平共享和利用率[14]，降低了云计算环境中使用GPU的成

本. NVIDIA 的多进程服务(Multi-Process Service, MPS)[15]是一种通过客户端-服务器架构实现的CUDA API的替代实现,旨在透明地支持多进程CUDA应用. 在最新NVIDIA GPU上利用Hyper-Q功能,从而提高GPU利用率并减少上下文切换开销. 此外,还有一些其他GPU共享方案的研究[16~19],进一步丰富了该领域的技术体系. vGPU的主流实现方案对比总结,如表1所示. 其中HAMi凭借部署便捷、功能完备的优势,通过CUDA API劫持与监控调节方法,分别实现了对显存与算力的细粒度切分,且作为该类资源切分型vGPU方案的典型代表,能够充分适配本文实验需求,故被选为研究中的vGPU实现方案.

表1　主流GPU资源共享方案对比总结

方案名称	实现方式	显存切分	算力切分
HAMi	显存隔离使用拦截CUDA API劫持的方法实现;使用监控调节方案实现GPU算力的切分,因此无法在短时间内限制算力,只能保证长时间的算力切分公平性	支持	支持
vCUDA	与HAMi的实现方式基本相同	不支持	不支持
cGPU	是最早使用内核劫持来实现GPU资源共享的方案,研发的cgpu_km组件可以对一个物理GPU虚拟出16个vGPU设备	支持	不支持
qGPU	类似于cGPU的GPU共享方案,实现了算力的强隔离	支持	支持
GPUshare	使用Linux cgroup、Docker限制和GPU插件,控制每个容器的显存/核数使用上限	支持	支持
NVIDIA MPS	在驱动层通过一个“共享守护进程”复用CUDA上下文,实现多个进程同时访问GPU	不支持	支持
NVIDIA MIG	基于硬件,直接在GPU层将SM、显存、缓存等切分成独立实例(MIG slices)	支持	支持
AdaGap	设计了基于深度q-network的自适应间隙感知资源分配策略,通过最小化异构集群中未充分利用的间隙来优化GPU使用	不支持	不支持
Antman	使用HAMi相同的技术实现了显存的切分.但是对算力的切分使用的操作符层级的API劫持,而不是内核级别的API劫持	支持	支持
rCUDA	同样是CUDA API劫持技术,但是实现了跨节点共同管理,支持对物理GPU的池化	不支持	不支持

vGPU技术因其细粒度资源划分和强安全隔离特性,已成为主流云服务商提供AI推理服务的首选基础设施. 与当前主流GPU资源共享方案对比,vGPU资源共享机制大多使用时间切片或简单的显存和算力切分技术,只实现了显存和算力能力的细粒度切分,而对底层的缓存、SM等资源并没有进行切分,导致在多任务并发执行时,不同任务之间可能会相互干扰[20],影响整体的资源利用效率和推理性能. 目前针对vGPU性能干扰的研究主要集中在干扰因素的分析上,如显存带宽竞争、计算单元抢占等,如何在大模型推理应用场景下对性能干扰精准预测,并据此优化资源配置的研究相对较少.

2.2　GPU资源共享干扰建模研究

为了应对GPU资源共享中的干扰问题,相关研究工作不断推进. Ayub等人[21]采用了DNN模型,对GPU中不同并发内核执行时产生的干扰进行建模和预测. 具体方法包括收集多种内核组合的性能数据,利用这些数据训练神经网络,从而捕捉不同任务间在共享缓存、计算单元和内存带宽上的竞争关系,以准确评估干扰对执行时间的影响,进而辅助调度优化. Xu等人[22]提出了在云计算中面向DNN模型推理场景的GPU资源分配方法,通过建立轻量级的DNN推理性能模型,捕捉GPU上共存推理工作负载之间的性能干扰,并基于此模型设计了一种成本效益高的GPU资源分配策略,以实现云环境中可预测的DNN推理性能.

一些研究者尝试采用机器学习技术建模. Xu等人[23]提出了一种基于机器学习的干扰感知虚拟机调度器,用于NVIDIA vGPU环境下的GPU资源共享. 通过深入研究vGPU的性能特性,构建了预测模型来量化共存虚拟机之间的干扰,并利用这些模型在集群级别调度虚拟机以最小化干扰,实验表明该方法能显著降低应用程序的运行时开销. 然而,这种方法主要关注虚拟化环境中的干扰问题,对于集群或裸金属环境的适用性有限. Wu等人[24]提出了透明GPU共享系统(Transparent GPU Sharing,TGS),这是一个为容器云中的深度学习训练工作负载提供透明GPU共享的系统. TGS通过自适应速率控制和透明统一内存管理机制,在操作系统层面实现了高GPU利用率和性能隔离,允许用户在容器中使用任何深度学习框架进行模型开发和训练,而无需对框架进行修改. Kim等人[25]提出了一种基于机器学习的干扰感知调度器,通过应用特征和资源使用记录来预测和避免GPU共享环境中的干扰,显著提高了任务完成时间和系统吞吐量.

一些研究者就特定环境展开研究. Geng等人[26]通过深入研究vGPU的性能特性,构建了预测模型来量化共存虚拟机之间的干扰,并利用这些模型在集群级别调度虚拟机以最小化干扰. 然而,这些研究主要集中在深度学习(Deep Learning,DL)工作负载上,这些工作负

载通常具有较为均匀的资源使用模式. 在实际应用中,资源使用模式可能更加复杂多样,从而限制了这些方法的普适性. Zhao 等人[27]提出了安全共享 GPU 集群框架,通过引入虚拟私有集群和多级单元结构,有效解决了深度学习训练中 GPU 共享集群的资源共享干扰问题,确保了资源共享的安全性,同时提高了集群利用率并减少了任务排队延迟. Chen 等人[28]提出了一种面向容器化集群的自适应 GPU 共享与调度方案,以应对多租户环境中复杂异构负载带来的资源竞争问题. 通过监测任务的资源利用率和运行状态,动态调整 GPU 分配策略,并设计了一个两阶段调度机制,分别在全局和本地层面优化任务分配与资源共享.

尽管已有研究在 GPU 资源共享干扰建模方面取得了一定进展,但仍存在许多挑战[29]. 首先,需要开发更全面的性能指标体系,以更准确地反映资源竞争和干扰的影响. 其次,应进一步探索在大模型推理负载应用场景下的资源干扰,以应对大模型负载日益增长的资源需求. 此外,还需要考虑如何将这些研究成果应用于更广泛的环境,包括裸金属服务器和异构计算环境. 最终目标是实现高效、可靠的 GPU 资源共享,以满足日益增长的计算需求.

2.3 GPU 资源分配方法研究

为了更好地提高 GPU 资源的利用率,研究者在 GPU 资源分配方法方面进行了大量研究[30]. 这些方法主要包括以下几个方向:首先,基于启发式算法的方法广泛应用于 GPU 资源分配问题中,如贪心算法[31]、局部搜索算法[32]、模拟退火算法[33](Simulated Annealing, SA)等. 这类方法通常通过局部搜索和启发式规则,快速生成较优解,以适应资源分配问题中的实时性需求. 其次,基于进化算法的资源分配方法也受到广泛关注,包括粒子群优化算法[34](Particle Swarm Optimization, PSO)、遗传算法[35](Genetic Algorithm, GA)、蚁群优化算法[36](Ant Colony Optimization algorithm, ACO)等. 这些方法通过模拟自然界群体行为或生物进化过程,实现了对大规模 GPU 资源调度问题的全局优化. 第三,基于强化学习的方法逐渐成为研究热点[37,38],通过学习环境状态与动作之间的关系,实现自适应的 GPU 资源分配策略. 此外,基于图模型和图神经网络的方法[39,40],通过建模任务和资源之间的依赖关系,以图结构表示资源调度问题,进一步提高了调度的智能化水平.

与此同时,针对 GPU 资源分配问题的实际应用场景,研究者也提出了多种优化目标与约束条件的组合模型. 例如,一些研究关注于在多租户共享 GPU 集群中,通过公平性约束、作业优先级管理和延迟敏感性优化,实现资源分配的多目标平衡[41];另一些工作则聚焦于异构 GPU 集群环境下的任务调度,解决不同 GPU 架构(如 A100、V100 等)之间的性能差异对资源分配效率的影响[42]. 此外,结合多任务学习、迁移学习等前沿技术,部分研究尝试提升资源调度算法的泛化能力和跨场景适应性[43]. 综上所述,GPU 资源分配方法的研究正逐渐从单一优化策略转向多目标、多约束、多场景下的综合优化方向,未来有望实现更高效、更智能的 GPU 资源管理体系.

3 vGPU 性能干扰建模

随着大模型推理服务需求的增长,推理系统正朝着高资源利用率和低延迟的目标演进发展. 大模型推理过程通常可分解为 2 个主要阶段:上下文预填充(Prefill)与自回归解码(Decode). 这 2 个关键阶段在计算特征、资源需求及其对性能指标的影响方面存在显著差异. 当采用 vGPU 技术将多路推理负载整合到同一张物理 GPU 卡上时,这种资源共享模式会引发显著的性能干扰,进而影响推理负载的 SLO.

3.1 大模型推理流程的关键阶段性能干扰分析

大模型推理的 2 个关键阶段示意图如图 2 所示. 在 Prefill 阶段,模型首先接收并处理输入的提示文本,通过词嵌入将其转换为向量表示. 随后利用 Transformer 结构计算所有输入词元(Token)的中间状态,最终生成首个输出 Token. 该阶段的核心计算特征为高度并行化的矩阵-矩阵运算,能充分调用 GPU 的计算能力,属于典型的计算密集型任务,其执行效率直接决定首 Token 响应时间(Time To First Token, TTFT)指标. 进入 Decode 阶段后,模型基于 Prefill 阶段生成的键值缓存(KV Cache),以自回归方式逐个生成后续 Token. 每次生成新 Token 时,系统仅需执行轻量级的矩阵-向量运算,同时动态更新 KV Cache 以支持后续生成. 由于该过程严格依赖历史 Token 序列且需串行执行,其性能主要受限于显存带宽而非计算能力,因此归类为内存密集型任务,其吞吐效率直接影响平均单 Token 生成时间(Time Per Output Token, TPOT)指标. 两阶段在计算模式、资源需求及性能瓶颈(TTFT、TPOT)上的显著差异,是优化大模型推理架构的关键依据.

在使用 vGPU 技术将多个推理负载部署于同一物

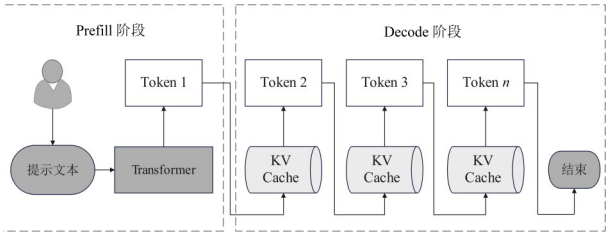


图 2 大模型推理 Prefill 阶段和 Decode 阶段示意图

理GPU的场景中,多个vGPU实例将共享底层的SM、带宽、二级缓存(L2 Cache)及功耗等关键资源.此类资源的竞争与共享,极易导致显存带宽冲突、SM调度延迟及缓存抖动等现象[44],引发推理性能的显著退化.对于Prefill阶段而言,由于其高度依赖并行计算资源,干扰主要表现为SM资源分配不足、CUDA调度拥塞以及核心频率下降,直接导致TTFT指标升高.而对于Decode阶段则受限于其对缓存与显存带宽的敏感性,特别是在高并发条件下,L2 Cache命中率降低、带宽抖动及功耗调度策略变化都会显著增加访问延迟,进而导致输出TPOT升高,且误差呈动态抖动特性[45].

影响大模型推理性能的干扰因素众多,如表2所提到的GPU硬件配置、推理负载特征、vGPU资源占比、实时监控信息等,都会影响大模型推理过程中的关键参数.下面将详细分析在vGPU环境下,各个干扰因素对大模型推理性能的影响.

3.2　大模型推理场景下vGPU性能干扰预测模型

本节将设计大模型推理场景下的vGPU性能干扰预测模型,用于预测推理过程TTFT和TPOT2个关键服务指标.相关的符号及其含义汇总在表2中.

表2　符号及含义

符号	含义与解释
$\mathcal{T}$	推理负载集合
$\mathcal{G}$	物理GPU集合
$M_{\text{total}}^{(g)}$	GPU g的总显存容量(GB)
$F_{\text{total}}^{(g)}$	GPU g的总计算能力(FLOPS)
$b_{\text{mem}}^{(g)}$	GPU g的显存带宽(GB/s)
c_g	GPU g的单位使用成本(元/小时)
$\text{TH}^{(t)}$	推理负载t的吞吐量
$R^{(t)}$	推理负载t承担请求的到达率
$I^{(t)}$	推理负载t承担请求的总输入Token数量
$O^{(t)}$	推理负载t承担请求的总输出Token数量
$\tau_{\text{ttft}}^{(t)}$	推理负载t承担请求的最大允许TTFT(ms)
$\tau_{\text{tpot}}^{(t)}$	推理负载t承担请求的最大允许TPOT(ms)
$B^{(t)}$	推理负载t的批处理大小(Batch Size)
$\text{SM}^{(t)}$	推理负载t占用显存经验值(GB)
K	GPU上已有的vGPU总数
$r_k^{(g)}$	第k个vGPU在GPU g上的资源占比
$M_k^{(g)}$	第k个vGPU的显存容量(GB)
$F_k^{(g)}$	第k个vGPU的算力能力(FLOPS)
$p_{\text{inst}}^{(g)}$	GPU g的瞬时功率(W)
$u_s^{(g)}$	GPU g的瞬时SM利用率(%)
$u_m^{(g)}$	GPU g的瞬时显存利用率(%)
$f_{\text{inst}}^{(g)}$	GPU g的瞬时频率(Hz)
x_{tkg}	推理负载t是否分配给GPU g的第k个vGPU
$\text{TTFT}_{\text{P99}}^{(t,k,g)}$	推理负载t在vGPU k上执行推理时的P99首Token响应时间(ms)
$\text{TPOT}_{\text{P99}}^{(t,k,g)}$	推理负载t在vGPU k上执行推理时的P99单Token生成时间(ms)
t_g	推理负载t在vGPU k上执行推理时的总推理时间

3.2.1　资源分配建模

每个vGPU的显存容量由GPU总显存和资源占比决定,如式(1)所示:

$$M_k^{(g)}=r_k^{(g)}\cdot M_{\text{total}}^{(g)} \tag{1}$$

其中,$r_k^{(g)}\in(0,\ 1]$且满足$\sum_{k=1}^{K_g} r_k^{(g)}\leqslant 1$.

每个vGPU的算力能力由GPU总算力和资源占比决定,如式(2)所示:

$$F_k^{(g)}=r_k^{(g)}\cdot F_{\text{total}}^{(g)} \tag{2}$$

通过5.2节中的性能干扰测试,发现在vGPU支持大模型推理的场景下,物理GPU的瞬时频率变化较为平缓.由此说明,在该场景下GPU的瞬时频率$f_{\text{inst}}^{(g)}$对性能干扰影响较小,因此在性能干扰建模过程中,不再引入$f_{\text{inst}}^{(g)}$参数.

3.2.2　推理负载建模

在大模型推理服务中,推理负载的批处理大小$B^{(t)}$

受任务到达率$R^{(t)}$、服务质量指标(如$\tau_{\text{ttft}}^{(t)}$、$\tau_{\text{tpot}}^{(t)}$)以及硬件资源限制等因素影响. 近年来,学术界针对推理任务的批处理大小设置问题进行了广泛研究. 多数研究工作将请求到达率作为批处理决策的重要输入参数,以保障SLO的满足[46,47]. 相关研究表明,为应对到达率的动态变化,需设计相应的批处理调度与自适应调整策略,以在性能与时延之间实现平衡. 同时,已有研究结果验证了请求到达率增大时,批处理规模相应增大的规律性关系[47]. 因此,为简化建模和分析,本文将批处理大小直接设置为与到达率相同,即$B^{(t)}=R^{(t)}$.

输入Token $I^{(t)}$通过专用分词器将提示词转换为Token序列并精确计算. 而推理工作负载在GPU设备上单独运行所需要分配的资源占比,与大模型的参数量、训练框架、训练精度、部署方式等变量都有非常大的关系. 为了简化模型的参数和关系,本文将结合HunggingFace提供的开源库Accelerate[48]与模型运行参数来完成$\text{SM}^{(t)}$的估算.

3.2.3 推理阶段关键指标模型

在3.1节中详细讨论了大模型推理过程的2个关键阶段,且2个关键阶段分别影响TTFT和TPOT这2个关键服务指标的服务质量. 因此,需要分别对2个关键阶段进行分析及建模.

在Prefill阶段以矩阵运算为主,属于计算密集型任务. 在vGPU推理环境下,TTFT主要受限于算力资源和CUDA调度性能,受到以下变量影响:输入Token数$I^{(t)}$、批处理大小$B^{(t)}$、vGPU分配算力$F_k^{(g)}$、SM利用率$u_s^{(g)}$、模型参数量$m^{(t)}$. 因此,TTFT的预测模型可表示为式(3):

$$\text{TTFT}(t,k,g)=\text{base}(t,k,g)_{\text{TTFT}}+\Delta^{\text{TTFT}} \tag{3}$$

其中,$\text{base}(t,k,g)_{\text{TTFT}}$定义为式(4):

$$\text{base}(t,k,g)_{\text{TTFT}}=\frac{\gamma_0+\gamma_1 B^{(t)}+\gamma_1\left(B^{(t)}\right)^2+\gamma_3 B^{(t)}m^{(t)}}{1+\gamma_4 u_s^{(g)}+\gamma_5\dfrac{m^{(t)}}{F_k^{(g)}}} \tag{4}$$

多模型干扰修正项Δ^{TTFT}定义为式(5):

$$\Delta^{\text{TTFT}}=\begin{cases}0, & \sum\limits_{t\in\mathcal{T}}x_{\text{tkg}}\leqslant 1\\ \gamma_6 B^{(t)}\left(1+\sum\limits_{t\in\mathcal{T}}x_{\text{tkg}}\right)^2, & \text{其他}\end{cases} \tag{5}$$

这里,x_{tkg}表示推理负载t是否正在vGPU设备上运行,其定义为式(6):

$$x_{\text{tkg}}=\begin{cases}1, & \text{推理负载}t\text{的}r_k^{(g)}>0\\ 0, & \text{推理负载}t\text{的}r_k^{(g)}=0\end{cases} \tag{6}$$

在Decode阶段为带宽、缓存敏感型,性能受限于内存带宽. 在推理环境下,TPOT的主导因子如下:批处理大小$B^{(t)}$、vGPU分配算力$F_k^{(g)}$、显存利用率$u_m^{(g)}$、模型参数量$m^{(t)}$. TPOT的预测模型可以表示为式(7):

$$\text{TPOT}(t,k,g)=\text{base}(t,k,g)_{\text{TPOT}}+\Delta^{\text{TPOT}} \tag{7}$$

其中,$\text{base}(t,k,g)_{\text{TPOT}}$定义为式(8):

$$\text{base}(t,k,g)_{\text{TPOT}}=\frac{\beta_0\left(B^{(t)}\right)^{\beta_1}\left(F_k^{(g)}\right)^{\beta_2}\left(m^{(t)}\right)^{\beta_3}}{1+u_m^{(g)}}+\beta_4 \tag{8}$$

此外,为了描述多模型在同一vGPU上运行时的干扰效应,引入修正项Δ^{TPOT},其定义为式(9):

$$\Delta^{\text{TPOT}}=\begin{cases}0, & \sum\limits_{t\in\mathcal{T}}x_{\text{tkg}}\leqslant 1\\ \beta_5 B^{(t)}\left(1+\sum\limits_{t\in\mathcal{T}}x_{\text{tkg}}\right), & \text{其他}\end{cases} \tag{9}$$

其中,x_{tkg}表示推理负载t是否运行在vGPU上. 当并发推理负载数目超过1时,该修正项用于补偿多任务并行带来的额外延迟.

基于以上模型定义,vGPU环境下的大模型推理性能干扰主要跟以下参数有关,其中包括5个推理负载参数(即$I^{(t)}$、$B^{(t)}$、$m^{(t)}$、$\tau_{\text{ttft}}^{(t)}$、$\tau_{\text{tpot}}^{(t)}$),3个vGPU配置参数(即$r_k^{(g)}$、$F_k^{(g)}$、$M_k^{(g)}$),2个监控参数(即$u_s^{(g)}$、$u_m^{(g)}$),3个GPU硬件配置参数(即$M_{\text{total}}^{(g)}$、$F_{\text{total}}^{(g)}$、$b_{\text{mem}}^{(g)}$). 模型中的γ_0~γ_6、β_0~β_5项可通过5.2节的测试数据,使用非线性最小二乘法进行拟合获得.

由于3个vGPU配置参数可以根据3.2.1的公式进行相互转化,所以在给定推理负载任务的TTFT和TPOT最大允许值后,就可以计算vGPU的配置参数以及批处理设置大小. 同样也可以使用vGPU配置参数和批处理大小来计算TTFT和TPOT的值,极大地减少了计算量.

3.3 成本优先的优化目标函数设计

基于以上大模型推理场景下vGPU性能干扰预测模型,将GPU资源分配优化目标定义如下:在满足请求达到率、TTFT、TPOT指标的情况下,如何为每个大模型推理负载任务t提供批处理大小$B^{(t)}$和切分比例为$r_k^{(g)}$的vGPU实例,实现优化分配GPU资源的货币成本.

3.3.1 决策变量

(1)$x_{\text{tkg}}\in\{0,1\}$:任务t是否分配给GPU g的第k个vGPU.

(2)$r_k^{(g)}\in(0,1)$:第k个vGPU在GPU g上的资源占比.

(3)$B^{(t)}$:当前推理负载的批处理大小.

3.3.2 约束条件

(1)任务分配约束. 每个任务必须被分配到某个vGPU:$\forall t\in\mathcal{T},\exists g\in\mathcal{G},k\in K_g: x_{\text{tkg}}=1$.

(2)性能SLOs约束. 满足TTFT和TPOT的最大允许值:$\text{TTFT}^{(t,k,g)}\leqslant\tau_{\text{ttft}}^{(t)}$, $\text{TPOT}^{(t,k,g)}\leqslant\tau_{\text{tpot}}^{(t)}$.

(3)显存限制约束. 每个任务的显存需求必须小于等于vGPU的可用显存:$\text{SM}^{(t)}\leqslant M_k^{(g)}$.

(4)吞吐量约束. 每个推理负载的吞吐量能够满足其到达率:$\mathrm{TH}^{(t)} \geqslant R^{(t)}$.

3.3.3 优化目标函数

$$\min \sum_{g \in \mathcal{G}} c_g t_g \sum_{k=1}^{K_g} r_k^{(g)} = \min \sum_{g \in \mathcal{G}} c_g \cdot \left(\mathrm{TTFT}^{(t,k,g)} + O^{(t)} \mathrm{TPOT}^{(t,k,g)}\right) \sum_{k=1}^{K_g} r_k^{(g)} \tag{10}$$

其中,优化目标函数式(10)中,输出Token数$O^{(t)}$是未知变量. 因此,可简化优化目标如下:在满足TTFT、TPOT这2个服务指标及其他约束条件的情况下,优化因性能干扰而导致的GPU资源分配. 这是一个多目标优化问题,往往不存在单一的最优解. 因此,本文将设计一种启发式算法,以解决在vGPU性能干扰的情况下对GPU资源的分配问题.

4 基于干扰感知的GPU资源配置方法

4.1 基于干扰感知的GPU资源配置方法设计

基于第3节的模型以及优化目标函数,我们提出了一种基于干扰感知的GPU资源分配方法,以实现对vGPU占用物理GPU资源的比例动态调控,从而有效支持大模型在共享GPU环境下的高效运行. 伪代码如算法1所示.

具体来说,给定一组大模型推理负载,以及它们允许的最大TTFT($\tau_{\mathrm{ttft}}^{(t)}$)、最大TPOT($\tau_{\mathrm{tpot}}^{(t)}$)和请求到达率$R^{(t)}$. 为了提高资源分配效率,本算法采用基于HuggingFace模型内存分析工具的预评估机制,将每个推理负载t所需的最小GPU资源比例$r_{\min}^{t}$作为初始分配基准,而非从0开始逐步递增的资源分配方式. 这一优化策略显著减少了资源逐步增加的次数,同时确保了初始分配满足模型运行的基本内存需求. 为避免资源碎片化,提高GPU资源利用率,根据vGPU资源需求对推理负载进行降序排序. 对于每个待分配负载w,遍历现有候选GPU设备(初始M=1),在保证当前物理GPU已分配的资源不超过其最大容量($\sum_{t \in \mathcal{T}} r_{\mathrm{ta}}^{(g)} \leqslant 1$)的前提下,通过迭代调整机制动态分配资源:当检测到当前负载的TTFT和TPOT超过阈值($\tau_{\mathrm{ttft}}^{(t)}$和$\tau_{\mathrm{tpot}}^{(t)}$)时,按$\Delta r$增量增加vGPU资源占比,直至满足服务质量要求. 同时引入资源增量约束$\mathrm{increase}_{\min}$来优化分配决策,在满足服务质量且因干扰引起的资源增量最小的GPU设备上分配支持推理负载w的vGPU资源. 若现有设备无法容纳新负载,则扩展新的GPU设备(M=M+1). 该算法通过这种需求驱动的启发式资源分配策略,不涉及多轮迭代求解的过程,实现了在有限GPU资源下的高效多任务调度,同时确保各推理负载的延迟约束得到满足.

算法1　资源分配方法

输入:每个推理负载$t \in \mathcal{T}$的请求到达率$R^{(t)}$,最大允许TTFT-τ_{ttft}^{t},最大允许TPOT-$\tau_{\mathrm{tpot}}^{(t)}$

输出:推理负载t的批处理大小$B^{(t)}$,支持推理负载t且不违反用户需求的vGPU占物理GPU的比例$r_t^{(g)}$

1. 初始化:根据HuggingFace模型占用内存工具得到推理负载t所需GPU,计算运行推理负载所需的vGPU资源占物理GPU的最小比例$r_{\min}^{t}$,$M=1$
2. 根据支持推理负载t的vGPU在物理GPU上的资源占比降序排序
3. for 所有推理负载w do
4. 　初始化推理负载w分配在j=−1的物理GPU上,表示其未分配到资源
5. 　for 所有GPU设备$g \in [1, M]$ do
6. 　　$r_{\mathrm{aw}}^{(g)}$=$r_{\min}^{w}$,是否重新分配资源f=true,放置推理负载引起的最小资源比例增加量$\mathrm{increase}_{\min}$=1
7. 　　while $\sum_{t \in \mathcal{T}} r_{\mathrm{ta}}^{(g)} \leqslant 1$ && f=true
8. 　　f=false
9. 　　for 所有在GPU g执行的推理负载t do
10. 　　根据3.2.3节TTFT和TPOT预测公式计算$\mathrm{TTFT}_t^{(g)}$和$\mathrm{TPOT}_t^{(g)}$
11. 　　if $\mathrm{TTFT}_t^{(g)} > \tau_{\mathrm{ttft}}^{t}$ && $\mathrm{TPOT}_t^{(g)} > \tau_{\mathrm{tpot}}^{(t)}$
12. 　　增加支持推理负载t的vGPU占物理GPU的比例$r_{\mathrm{ta}}^{(g)} \leftarrow r_{\mathrm{ta}}^{(g)} + \Delta r$
13. 　　f=true
14. 　　end if
15. 　　end for
16. 　end while
17. 　计算GPU分配比例增加量$\mathrm{increase}_t^{(g)}$,$t \in \mathcal{T}$
18. 　if $\sum_{t \in \mathcal{T}} r_{\mathrm{ta}}^{(g)} \leqslant 1$ && $\sum_{t \in \mathcal{T}} \mathrm{increase}_t^{(g)} \leqslant \mathrm{increase}_{\min}$ then
19. 　　将推理负载w放置在设备g上j=g
20. 　end if
21. end for
22. if 当前所有GPU设备无法满足j=−1
23. 　增加GPU设备M=M+1,将推理负载w放置在新增的GPU上$r_{\mathrm{wa}}^{(M)} \leftarrow r_{\min}^{w}$
24. else
25. 　更新$r_{\mathrm{ta}}^{(g)}$,$t \in \mathcal{T}$
26. end if
27. end for

4.2 基于vGPU环境的大模型推理负载支撑框架实现

为保证基于干扰感知的GPU资源配置方法可以自动化运行,本文设计了基于vGPU环境的大模型推理负载生成、性能预测、放置策略生成、监控信息获取的完整解决方案. 基于vGPU环境的大模型推理负载支撑解决方案如图3所示.

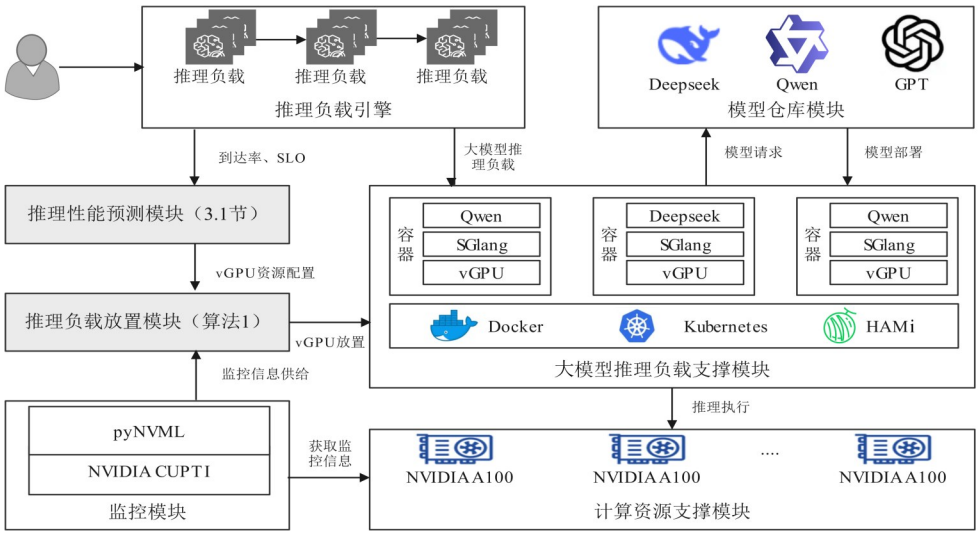


图3 基于vGPU环境的大模型推理负载支撑框架

从图3中可以看出，本研究构建的vGPU性能测试框架包含6大核心模块.

推理负载引擎. 基于SGlang的基准测试脚本，设计了一套灵活的推理负载引擎. 该引擎能够自动化生成独占执行和并行执行2种模式的推理负载，并为推理负载配置不同的运行参数，为多样化的大模型推理性能测试与验证提供了灵活可配置的自动化支持.

推理性能预测模块和推理负载放置模块. 使用Python、Shell等语言开发，总代码超过3 000行，所涉及代码和数据已经在Github开源，访问地址为https://github.com/ICALab-nsccjn/vGPU-LLM-Allocator.git. 其功能实现了接收推理引擎发送的推理负载运行指令，并且使用3.2节中的性能预测模型，生成vGPU的配置方案. 基于vGPU的配置方案，使用4.1节的资源分配算法，将vGPU放置到局部最优的物理GPU上运行.

大模型推理负载支撑模块. 采用云原生技术栈构建，接收推理负载放置模块的资源放置指令，为大模型推理负载提供了灵活高效的运行环境. 在底层架构中，使用Kubernetes作为核心的资源调度和任务编排平台，结合HAMi vGPU解决方案实现GPU资源的虚拟化管理. HAMi作为业界领先的vGPU实现方案，支持对GPU算力和显存进行百分比级别的细粒度切分与隔离，提供了更精细的资源划分能力（最小可支持1%的GPU资源分配），这使得系统能够更好地适应多租户、多任务场景下的资源分配需求. 在大模型推理支持方面，选用SGlang[49]作为核心推理引擎，该组件不仅具备与TensorRT-LLM相当的性能表现，而且通过解耦对特定GPU硬件的依赖，使得模型能够无缝迁移到不同GPU平台进行测试和验证，极大地提升了系统的可移植性和实验效率.

模型仓库模块. 为了降低因不同大模型架构差异导致的推理延迟影响，基于SGlang的推理引擎构建了一个统一的大模型仓库，可以为每个大模型提供标准化的加载和推理接口，确保不同架构的模型能够无缝集成. 该模型仓库已经集成了多种主流开源大模型，包括DeepSeek、Qwen、LLama等，有效支撑了第5节中对不同大模型推理过程的负载监控.

计算资源支撑模块. 为大模型推理负载的硬件支撑平台，最终所有的大模型推理负载会运行计算资源支撑模块. 因此，其性能表现直接影响模型推理的效率和稳定性. 为全面评估多个大模型推理负载在共享GPU环境中的性能干扰效应，构建了基于高性能物理服务器的单节点计算集群. 该服务器配备8块NVIDIA A100 40 GB GPU，采用NVIDIA Ampere架构[50]，具有512 GB内存，确保单GPU可以支持多个中等规模大模型推理负载. 通过这一硬件配置，能够模拟真实数据中心环境中多个大模型并行推理的场景，深入分析GPU资源竞争、功耗及显存带宽瓶颈对推理性能的影响，为优化大模型推理负载的资源分配策略提供可靠依据.

监控模块. 构建了多层次、全方位的资源监控体系，实现了从物理GPU到推理负载的全面监控覆盖. 在GPU资源监控层面，采用pyNVML工具进行实时数据采集，该工具能够以秒级粒度持续记录GPU利用率、显存占用、温度、功耗等关键监控指标，并通过对监控数据的持久化存储，为干扰分析及预测提供支撑；在推理负载监控方面，采用NVIDIA CUPTI工具，监控推理负载执行过程中的动态随机缓存器利用率、GPU活动时间等关键监控指标，实现了完善的推理负载监控数据采集和分析体系.

5　实验验证

5.1　实验环境

(1)测试集群. 如4.2节所述,本文搭建了一台包含8块NVIDIA A100的GPU,512 GB内存,2块Intel(R) Xeon(R) Platinum 8358 CPU. 根据3.1节构建的性能干扰预测模型要求,需要获取GPU的物理硬件信息. NVIDIA A100的算力为312 TFlops(FP16),额定功率为400 W,最大频率为1.4 GHz,内存带宽为1 555 GB/s,显存容量为40 GB,SM个数为108个. 另外,为验证预测模型的泛化性,搭建了一台配备2块NVIDIA RTX6000 GPU、192 GB内存以及64核基于Intel Xeon Cascade Lake架构的虚拟CPU的计算环境.

(2)软件平台. 使用了Kubernetes 1.30.4、Docker 20.10.8、HAMi 2.4.1、SGlang. NVIDIA驱动版本和CUDA版本分别为550.127.08和12.4.

(3)工作负载. 本文为了增加预测模型和资源分配方法的普适性,选取了DeepSeek、Qwen这2类模型,包含6种参数的大模型,分别是Qwen3-0.6B-Base、DeepSeek-R1-Distill-Qwen-1.5B、Qwen3-1.7B-Base、Qwen3-4B-Base、DeepSeek-R1-Distill-Qwen-7B和Qwen3-8B-Base.

(4)基线和指标. 目前学术界较少对vGPU支撑大模型推理负载进行深入研究,更多的是对深度学习模型在GPU资源共享环境下的研究,这也体现了本文的创新性. 本文将设置2类测试,分别是预测误差测试和资源占用测试. 在预测误差测试中,不设置比较基线,只进行性能干扰预测模型应用于不同模型和场景时,预测误差指标的横向对比;在资源占用测试中,将对比本文的资源分配方法,默认资源分配方案在满足TTFT和TPOT服务指标的情况下,资源使用的指标.

5.2　实验数据构建及预测误差测试

目前没有公开的基于vGPU环境下,不同大模型推理负载的服务指标测试数据集. 因此,为了给3.2节构建的vGPU性能干扰预测模型提供足够的数据来拟合未知参数,基于5.1节构建的NVIDIA A100 GPU实验环境和大模型,进行了3类78组实验. 具体来说,第1类实验,每次单独创建1个容器并启动1个模型服务,每个容器分别配备10%、20%、30%、40%、50%、60%、70%、80%、90%、100%的GPU资源,所部署的模型包括Qwen3-0.6B-Base、DeepSeek-R1-Distill-Qwen-1.5B、Qwen3-1.7B-Base、Qwen3-4B-Base、DeepSeek-R1-Distill-Qwen-7B和Qwen3-8B-Base. 第2类实验,每个模型分4组,分别同时创建1~4个容器并在每个容器中启动1个模型服务,每个容器配备25%GPU资源,分别部署了Qwen3-0.6B-Base、DeepSeek-R1-Distill-Qwen-1.5B和Qwen3-1.7B-Base这3种模型. 第3类实验,每个模型分2组,分别同时创建1~2个容器,每个容器配备50%GPU资源,分别部署了Qwen3-4B-Base、DeepSeek-R1-Distill-Qwen-7B和Qwen3-8B-Base这3种大模型. 每组实验都为模型推理负载设置了1、2、4、16和32的并发批处理变量,来捕获因模型内并发而带来的性能干扰.

监控模块会在推理过程中采集GPU,推理负载的关键监控指标. 每组实验共重复3次,使用3组平均值作为最终的监控数据,形成不同模型、不同资源分配和不同批处理大小情况下的多维指标数据集. 为简明起见,本文选取其中具有代表性的3组干扰实验结果进行展示,如图4~6所示.

从图4~6可知,在相同大模型推理场景下,当批处理大小与请求到达率相等时,批处理大小与TTFT呈显著正相关关系. 随着批处理规模的增大,TTFT逐渐升高,表明过大的批处理规模将导致模型响应延迟显著增加. 在GPU资源共享场景中,当同一GPU上并行部署多个相同的大模型实例时,TTFT随模型数量的增加而明显上升,且这种上升趋势在模型参数规模较大时

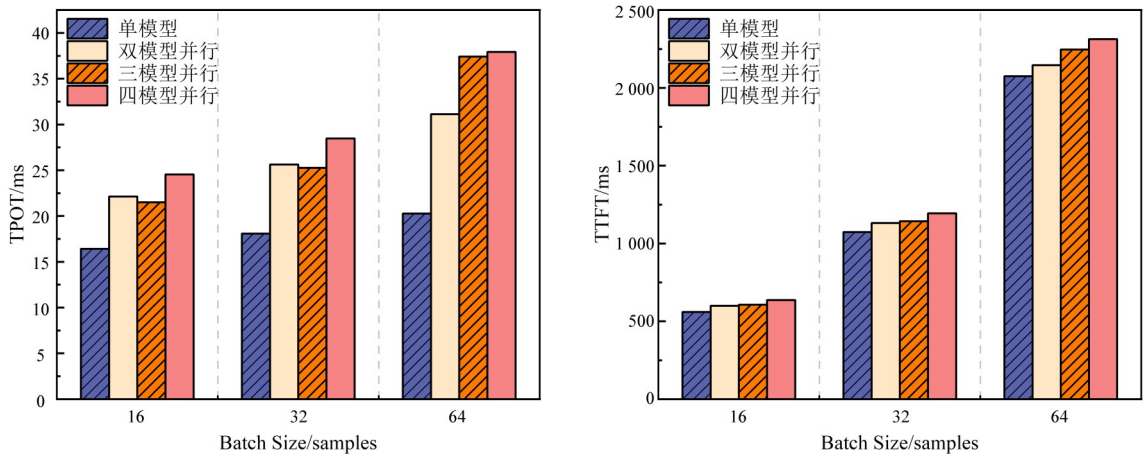


图4　Qwen3-0.6B-Base的干扰示意图

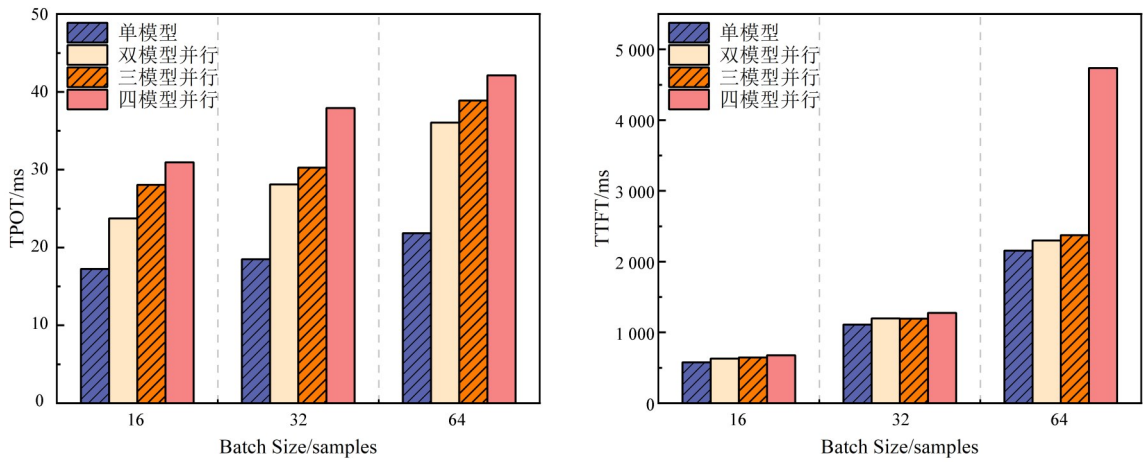


图5 Qwen3-1.7B-Base的干扰示意图

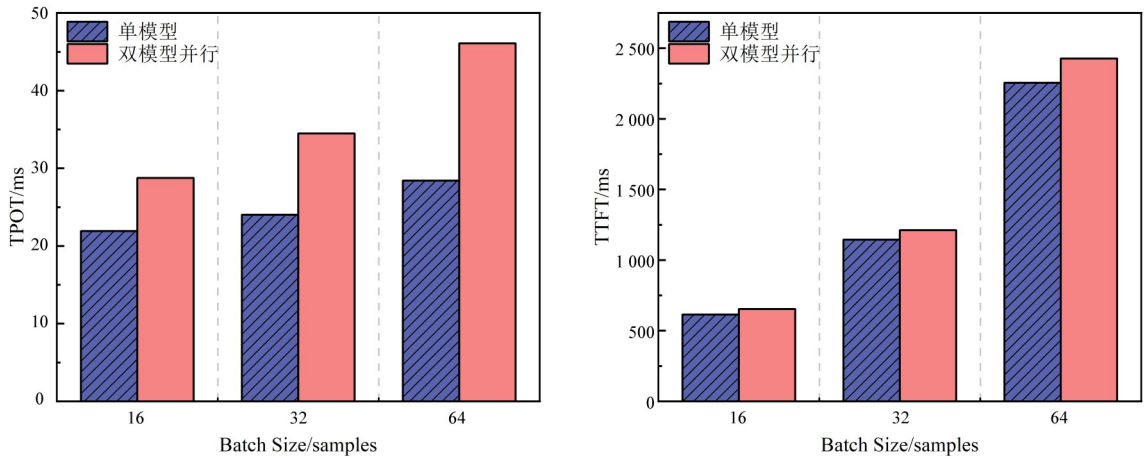


图6 Qwen3-4B-Base的干扰示意图

更加突出,反映出多实例部署下GPU内部资源争用对推理启动延迟的显著影响.

对于TPOT,在单模型场景中,其值随批处理大小的增大呈轻微上升趋势.这是由于批处理能力提升的同时,也引入了更强的资源竞争与模型内部调度复杂度增加.尽管TPOT略有上升,但总体增幅有限,从系统整体吞吐量优化的角度来看,该性能折中仍可接受.此外,当同一GPU内的模型部署数量增加时,TPOT亦呈上升趋势,进一步验证了多模型共享资源条件下推理阶段性能受到的负面影响.

此外,为验证预测模型的泛化性,本文在5.1节所述的NVIDIA RTX6000 GPU实验环境下复现了与上述设置一样的实验方案.选取Qwen3-0.6B-Base、DeepSeek-R1-Distill-Qwen-1.5B、Qwen3-1.7B-Base、Qwen3-4B-Base这4个模型,共进行了3类51组实验,相关实验数据也已公开到代码仓库中.

在完成实验数据集构建后,本文基于第3.2节提出的vGPU性能干扰预测模型,采用非线性最小二乘法进行拟合,获得了Prefill阶段和Decode阶段的计算系数.基于拟合完成的模型,本文通过调整vGPU资源占比、批处理大小及并发推理模型数量等参数,构建多种大模型推理负载运行场景.随后,根据实际运行结果与模型预测得到的TTFT和TPOT指标,对vGPU性能干扰预测模型的误差进行了评估.TTFT和TPOT的预测值与实测值散点图如图7和图8所示.

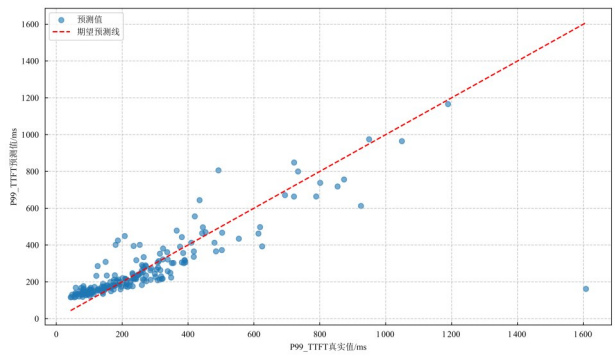


图7 TTFT预测值散点图(R^2=0.657 6)

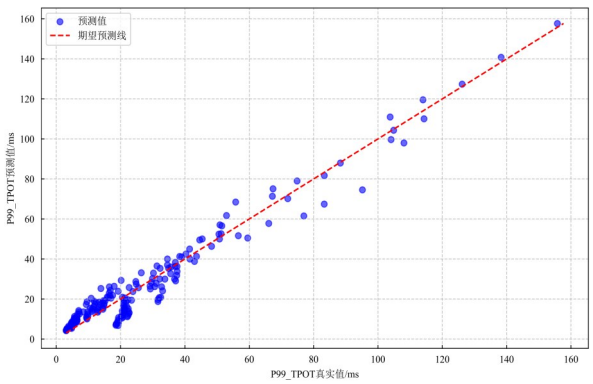


图8　TPOT预测值散点图(R^2=0.953 8)

结果表明,本文提出的vGPU性能干扰预测模型在NVIDIA A100 GPU环境下的TPOT指标拟合精度较高,R^2约为0.95,能够较为准确地刻画不同算力比例、批处理大小及并发数量下的吞吐性能变化规律. 相较之下,TTFT的R^2约为0.66,预测精度相对较低. 造成这一差异的原因在于,TTFT仅在推理任务的首次令牌生成阶段出现一次,其性能表现对系统瞬态状态(如模型加载延迟、缓存命中率、上下文切换及显存带宽竞争等因素)高度敏感. 在计算P99时,这些偶发性极值会被放大,导致TTFT的波动性进一步增强. 而TPOT作为连续输出阶段的统计平均指标,对单次异常更具鲁棒性,因此拟合效果更稳定、精度更高.

为进一步验证模型的泛化能力,本文在NVIDIA RTX6000 GPU环境下对同一模型进行了迁移测试. 结果显示,预测模型在TTFT与TPOT指标上的R^2分别达到0.65与0.85,与NVIDIA A100 GPU环境下的表现总体一致. 该结果表明,本文提出的vGPU性能干扰预测模型在不同硬件平台下均能保持较好的拟合效果,具备一定的跨平台泛化能力.

总体而言,该模型能够较好地反映Prefill与Decode阶段的性能变化趋势,并在主要性能指标上实现了较高的一致性. 该结果验证了本文提出的vGPU性能干扰预测模型在多模型并发推理场景下的有效性与可行性,为后续的资源分配优化与性能调度提供了一定的理论依据.

5.3　效果验证测试

为了验证本文所构建的基于vGPU性能干扰感知的大模型推理负载资源高效配置方法的有效性,选取了5个模型,构造了共计8种推理负载,各个推理负载的特征信息如表3所示. 由于目前学术界缺乏对vGPU环境下大模型负载配置资源的动态分配方法的研究,所以将本文提出的算法1与传统资源分配策略针对8种推理负载生成的配置方案进行了对比.

表3　推理负载及其特征信息

推理负载		DeepSeek-R1-Distill-Qwen-1.5B	Qwen3-1.7B-Base	Qwen3-4B-Base	DeepSeek-R1-Distill-Qwen-7B	Qwen3-8B-Base
S1	*B*/samples	8	2	4	8	5
	TTFT/ms	200	200	250	650	800
	TPOT/ms	50	40	50	100	70
S2	*B*/samples	16	—	—	9	—
	TTFT/ms	600	—	—	550	—
	TPOT/ms	60	—	—	100	—
S3	*B*/samples	32	—	—	—	—
	TTFT/ms	600	—	—	—	—
	TPOT/ms	80	—	—	—	—

传统策略基于HuggingFace显存分析工具估算模型基础显存需求,并在此基础上分别固定增加30%、50%显存,形成3组对比资源分配方案,分别记为传统策略、固定增加资源策略1、固定增加资源策略2. 相比之下,本文算法综合考虑了模型动态显存变化、批处理大小、TPOT/TTFT目标约束及多模型并发执行时潜在的vGPU性能干扰因素,从而生成了更紧凑高效的资源分配方案. 图9展示了4种策略下产生的具体GPU配置结果. 本文算法所生成的方案仅使用了4张物理GPU即可满足8个推理负载的资源需求并且保证满足性能约束. 而固定增加资源策略2在尽可能满足SLO条件下需要占用5张物理GPU. 因此,按物理GPU卡数量计算,资源成本节省高达20%. 同时,传统策略和固定增加资源策略导致的显存碎片大于本文方法导致的显存碎片. 在固定增加资源策略1的情况下,最大允许TTFT或最大允许TPOT的违规率约25%. 本文算法在物理GPU占用量、资源利用率、性能达标率及成本效益等方面均表现出显著优势. 该方法能够有效降低资源碎片化程度,为vGPU环境下的大模型推理服务提供高效、可靠的资源配置方案,带来可观的成本节约.

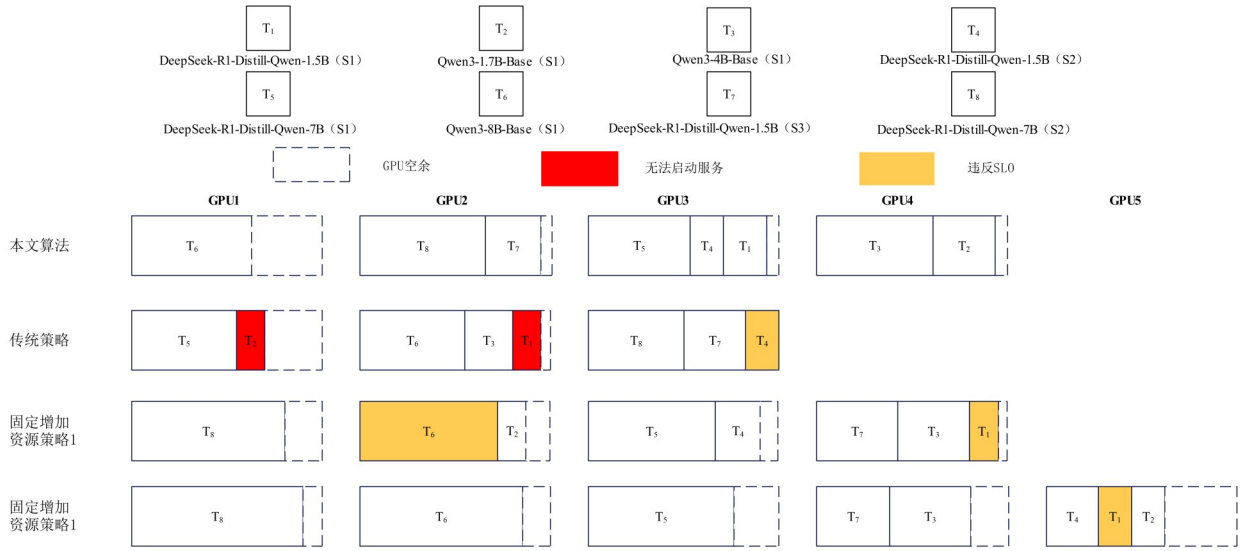


图9 有效性对比案例示意图

5.4 算法运行开销测试

为评估本文所提出资源分配算法的运行开销，从算法执行时间与内存消耗2个方面开展了实验测评．具体而言，在实验GPU服务器上运行本文设计的GPU资源分配算法，当负载数量为10时，算法执行时间为0.07 s，内存占用为93.38 MB；当负载数量增至100时，执行时间上升至8.55 s，内存占用为93.64 MB，资源分配算法的运行开销测试结果如图10所示．实验结果表明，随着负载规模的扩大，算法执行时间与内存使用量均呈总体线性增长趋势．考虑到大模型推理任务通常具有参数规模大、执行周期长等特性，属于持续运行而非短时释放的负载类型，因此，本文资源分配算法在实际生产环境中的运行开销处于可接受范围内．

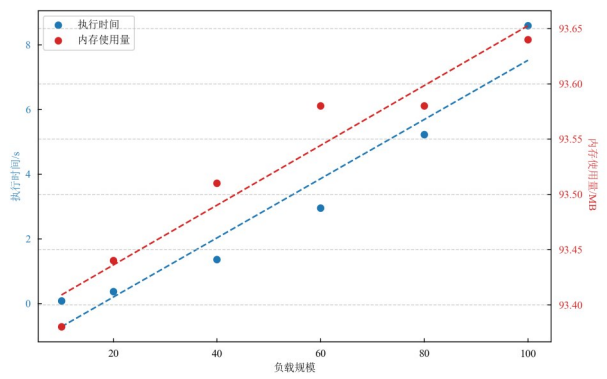


图10 资源分配算法的运行开销测试结果图

6 总结与展望

针对vGPU环境下大模型推理负载的多维性能干扰问题，本文从性能干扰感知角度出发，提出了一种基于vGPU性能干扰感知的GPU资源高效配置方法．本文融合了最新的大模型推理服务框架与组件，充分考虑真实业务场景中的推理服务需求，构建了基于已知变量的轻量化vGPU性能干扰预测模型，并设计了多目标优化函数与约束条件，以实现对资源分配的精细化调度与优化．大量实验结果表明，所提出的干扰预测模型能够基于硬件配置、推理负载特征及服务质量约束等多源信息，准确评估推理任务所需的vGPU资源规模．在严格满足SLO要求的前提下，本文提出的资源配置方法相比传统方案可降低超过20%的资源使用成本，验证了方法的有效性与经济性．

尽管本文取得了一定的研究成果，但仍存在以下2个方面的局限性：首先，目前所提出的性能干扰预测模型为静态离线模型，参数在拟合完成后固定，尚不支持在实际推理过程中进行在线自适应更新；其次，本文提出的资源配置算法主要基于启发式策略，虽然在实验中表现出较好的优化效果，但理论上仍可能存在更优的算法设计或搜索策略，理论性能上限有待进一步研究．

未来研究可从以下3个方向进一步深入：其一，构建多样化硬件环境，并验证本文方法在极端环境下的泛化性及稳定性；其二，探索更高效的优化策略与算法设计，以提升资源配置策略的全局优化能力；其三，突破现有数学建模的局限，探索基于深度学习或强化学习的动态干扰建模方法．通过数据驱动的方式增强模型对复杂vGPU环境的自适应能力，引入在线学习机制以增强模型的动态调整能力，从而推动vGPU技术在AI服务领域的深度应用．

综上，本文研究不仅在学术上具备前瞻性和创新性，也为云计算服务商在异构资源管理与大模型推理优化方面提供了可行的技术路径与实践参考．

参考文献

[1] SANDERSON K. GPT-4 is here: What scientists think[J]. Nature, 2023, 615(7954): 773.

[2] DENG Z H, MA W L, HAN Q L, et al. Exploring DeepSeek: A survey on advances, applications, challenges and future directions[J]. IEEE/CAA Journal of Automatica Sinica, 2025, 12(5): 872-893.

[3] 李晨, 刘畅, 葛一漩, 等. 多GPU系统非一致存储访问优化: 研究进展与展望[J]. 电子学报, 2024, 52(5): 1783-1800. LI C, LIU C, GE Y X, et al. Non-uniform memory access optimization on multi-GPU systems: Research progress and prospect[J]. Acta Electronica Sinica, 2024, 52(5): 1783-1800. (in Chinese)

[4] SONG Y X, MI Z Y, XIE H T, et al. PowerInfer: Fast large language model serving with a consumer-grade GPU[C]//Proceedings of the ACM SIGOPS 30th Symposium on Operating Systems Principles. New York: ACM, 2024: 590-606.

[5] BRIDGES R A, IMAM N, MINTZ T M. Understanding GPU power: A survey of profiling, modeling, and simulation methods[J]. ACM Computing Surveys, 2016, 49(3): 1-27.

[6] DHAKAL A, KULKARNI S G, RAMAKRISHNAN K K. GSLICE: Controlled spatial sharing of GPUs for a scalable inference platform[C]//Proceedings of the 11th ACM Symposium on Cloud Computing. New York: ACM, 2020: 492-506.

[7] CHOI S, LEE S, KIM Y, et al. Serving heterogeneous machine learning models on Multi-GPU servers with Spatio-Temporal sharing[C]//2022 USENIX Annual Technical Conference (USENIX ATC 22). California: USENIX Association, 2022: 199-216.

[8] ZHANG B W, LI S X, LI Z Z. MIGER: Integrating multi-instance GPU and multi-process service for deep learning clusters[C]//Proceedings of the 53rd International Conference on Parallel Processing. New York: ACM, 2024: 504-513.

[9] HAMi. Project-HAMi: Heterogeneous AI computing virtualization middleware[EB/OL]. [2024-09-29]. https://github.com/Project-HAMi/HAMi/tree/release-v2.4.

[10] SHI L, CHEN H, SUN J H, et al. vCUDA: GPU-accelerated high-performance computing in virtual machines[J]. IEEE Transactions on Computers, 2012, 61(6): 804-816.

[11] LIN Y S, LIN C Y, LEE C R, et al. qCUDA: GPGPU virtualization for high bandwidth efficiency[C]//2019 IEEE International Conference on Cloud Computing Technology and Science. Piscataway: IEEE, 2020: 95-102.

[12] GU J F, WANG P X, DAVID NÚÑEZ ARAYA I, et al. HAS-GPU: Efficient hybrid auto-scaling with fine-grained GPU allocation for SLO-aware serverless inferences[C]//Euro-Par 2025: Parallel Processing. New York: ACM, 2025: 159-174.

[13] ZHANG S L, XU A, CHEN Q, et al. Efficient performance-aware GPU sharing with compatibility and isolation through kernel space interception[C]//2025 USENIX Annual Technical Conference (USENIX ATC 25). California: USENIX Association, 2025: 1003-1019.

[14] GOSWAMI A, YOUNG J, SCHWAN K, et al. GPUShare: Fair-sharing middleware for GPU clouds[C]//2016 IEEE International Parallel and Distributed Processing Symposium Workshops. Piscataway: IEEE, 2016: 1769-1776.

[15] WEAVER A, KAVI K, MILOJICIC D, et al. Granularity- and interference-aware GPU sharing with MPS[C]//Proceedings of the SC'24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis. New York: ACM, 2025: 1630-1637.

[16] ZHANG Z K, ALLEN T, YAO F, et al. TunneLs for bootlegging: Fully reverse-engineering GPU TLBs for challenging isolation guarantees of NVIDIA MIG[C]//Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2023: 960-974.

[17] WANG S, CHEN S P, SHI Y M, et al. AdaGap: An adaptive gap-aware resource allocation strategy for GPU sharing in heterogeneous clusters[J]. Future Generation Computer Systems, 2025, 173: 107883.

[18] XIAO W C, REN S R, LI Y, et al. AntMan: Dynamic scaling on GPU clusters for deep learning[C]//14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). California: USENIX Association, 2020: 533-548.

[19] DUATO J, PEÑA A J, SILLA F, et al. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters[C]//2010 International Conference on High Performance Computing & Simulation. Piscataway: IEEE, 2010: 224-231.

[20] ZHAO C, GAO W, NIE F P, et al. Fair and cache blocking aware warp scheduling for concurrent kernel execution on GPU[J]. Future Generation Computer Systems, 2020, 112: 1093-1105.

[21] AYUB M, HELMY T. Concurrent kernel execution and interference analysis on GPUs using deep learning ap-

proaches[J]. Journal of King Saud University - Computer and Information Sciences, 2022, 34(10): 10193-10204.

[22] XU F, XU J N, CHEN J B, et al. iGniter: Interference-aware GPU resource provisioning for predictable DNN inference in the cloud[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(3): 812-827.

[23] XU X, ZHANG N, CUI M, et al. Characterization and prediction of performance interference on mediated passthrough GPUs for interference-aware scheduler[C]//Proceedings of the 11th USENIX Conference on Hot Topics in Cloud Computing. New York: ACM, 2019: 14.

[24] WU B Y, ZHANG Z L, BAI Z H, et al. Transparent GPU sharing in container clouds for deep learning workloads[C]// 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23). California: USENIX Association, 2023: 69-85.

[25] KIM S, KIM Y. Co-scheML: Interference-aware container co-scheduling scheme using machine learning application profiles for GPU clusters[C]//2020 IEEE International Conference on Cluster Computing. Piscataway: IEEE, 2020: 104-108.

[26] GENG X, ZHANG H T, ZHAO Z Y, et al. Interference-aware parallelization for deep learning workload in GPU cluster[J]. Cluster Computing, 2020, 23(4): 2689-2702.

[27] ZHAO H Y, HAN Z H, YANG Z, et al. HiveD: Sharing a GPU cluster for deep learning with guarantees[C]//14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20). California: USENIX Association, 2020: 515-532.

[28] CHEN Q C, OH J, KIM S, et al. Design of an adaptive GPU sharing and scheduling scheme in container-based cluster[J]. Cluster Computing, 2020, 23(3): 2179-2191.

[29] NA S, KIM J, LEE S, et al. Supporting secure multi-GPU computing with dynamic and batched metadata management[C]//2024 IEEE International Symposium on High-Performance Computer Architecture. Piscataway: IEEE, 2024: 204-217.

[30] GAO W, OUYANG Z Y, SUN P, et al. IceFrog: A layer-elastic scheduling system for deep learning training in GPU clusters[J]. IEEE Transactions on Parallel and Distributed Systems, 2025, 36(6): 1071-1086.

[31] TAO X R, PAN Q K, GAO L. An iterated greedy algorithm with reinforcement learning for distributed hybrid flowshop problems with job merging[J]. IEEE Transactions on Evolutionary Computation, 2025, 29(3): 589-600.

[32] ZHANG W Q, LI C, GEN M, et al. A multiobjective memetic algorithm with particle swarm optimization and Q-learning-based local search for energy-efficient distributed heterogeneous hybrid flow-shop scheduling problem[J]. Expert Systems with Applications, 2024, 237: 121570.

[33] KUMAR B A, JYOTHI B, SINGH A R, et al. Hybrid genetic algorithm-simulated annealing based electric vehicle charging station placement for optimizing distribution network resilience[J]. Scientific Reports, 2024, 14: 7637.

[34] GONG C, ZHOU N R, XIA S H, et al. Quantum particle swarm optimization algorithm based on diversity migration strategy[J]. Future Generation Computer Systems, 2024, 157: 445-458.

[35] LONDE M A, PESSOA L S, ANDRADE C E, et al. Biased random-key genetic algorithms: A review[J]. European Journal of Operational Research, 2025, 321(1): 1-22.

[36] ROSTAMI S, BROUMANDNIA A, KHADEMZADEH A. An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm[J]. The Journal of Supercomputing, 2024, 80(6): 7812-7848.

[37] WANG Y M, HAO M, HE H, et al. DRLCAP: Runtime GPU frequency capping with deep reinforcement learning[J]. IEEE Transactions on Sustainable Computing, 2024, 9(5): 712-726.

[38] LIU Z H, XU X, QIAO P, et al. Acceleration for deep reinforcement learning using parallel and distributed computing: A survey[J]. ACM Computing Surveys, 2025, 57(4): 1-35.

[39] 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]. The Journal of Supercomputing, 2024, 80(16): 24138-24172.

[40] LIU T F, CHEN Y R, LI D, et al. BGL: GPU-efficient GNN training by optimizing graph data I/O and preprocessing[EB/OL]. (2021-12-16)[2025-10-10]. https://arXiv.org/abs/2112.08541.

[41] MO Z Z, XU H L, LAU W C. Optimal resource efficiency with fairness in heterogeneous GPU clusters[C]//Proceedings of the 25th International Middleware Conference. New York: ACM, 2024: 36-48.

[42] WANG S, CHEN S P, SHI Y M. GPARS: Graph predictive algorithm for efficient resource scheduling in heterogeneous GPU clusters[J]. Future Generation Computer Systems, 2024, 152: 127-137.

[43] ARIMA E, KANG M, SABA I, et al. Optimizing hardware resource partitioning and job allocations on modern

GPUs under power caps[C]//Proceedings of the 51st International Conference on Parallel Processing. New York: ACM, 2023: 1-10.

[44] STRATI F, MA X Z, KLIMOVIC A. Orion: Interference-aware, fine-grained GPU sharing for ML applications[C]// Proceedings of the Nineteenth European Conference on Computer Systems. New York: ACM, 2024: 1075-1092.

[45] LEE W, LEE J, SEO J, et al. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management[EB/OL]. (2024-06-28) [2025-10-10]. https://arXiv.org/abs/2406.19707.

[46] ALI A, PINCIROLI R, YAN F, et al. BATCH: Machine learning inference serving on serverless platforms with adaptive batching[C]//SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. Piscataway: IEEE, 2021: 1-15.

[47] MIAO X P, SHI C N, DUAN J F, et al. SpotServe: Serving generative large language models on preemptible instances[C]//Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. New York: ACM, 2024: 1112-1127.

[48] WALKOWIAK B, WALKOWIAK T. Assessing inference time in large language models[M]//System Dependability-Theory and Applications. Cham: Springer, 2024: 296-305.

[49] BARRETT C, CAO S Y, GONZALEZ J, et al. SGLang: Efficient execution of structured language model programs[EB/OL]. (2024-06-06)[2025-10-01]. https://arxiv.org/abs/2312.07104.

[50] LUO W L, FAN R B, LI Z Y, et al. Benchmarking and dissecting the NVIDIA hopper GPU architecture[C]// 2024 IEEE International Parallel and Distributed Processing Symposium. Piscataway: IEEE, 2024: 656-667.

作者简介

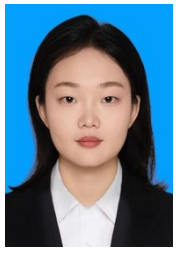
张　虎　男,1987年1月出生于山东省济南市. 现为山东省计算中心(国家超级计算济南中心)副研究员. 主要研究方向为大模型推理优化、算力互联网.
E-mail: zhanghu@sdas.org

孙明辉　男,1997年6月出生于山东省泰安市. 现为山东省计算中心(国家超级计算济南中心)工程师. 主要研究方向为算力资源调度.
E-mail: sunminghui9999@163.com

刘　杨　男,1984年6月出生于黑龙江省哈尔滨市. 现为北京邮电大学教授. 主要研究方向为算力芯片与网络. 中国电子学会会员编号:E190022238M.
E-mail: liu.yang@bupt.edu.cn

戴鸿君　男,1981年5月出生于山东省泰安市. 现为山东大学教授. 主要研究方向为计算机体系结构.
E-mail: dahogn@sdu.edu.cn

王继彬　男,1984年5月出生于山东省临沂市. 现为山东省计算中心(国家超级计算济南中心)研究员. 主要研究方向为虚拟资源调度.
E-mail: wangjb@sdas.org



张有利　女,2000年4月出生于山东省菏泽市. 现为齐鲁工业大学(山东省科学院)硕士研究生. 主要研究方向为异构计算环境下的任务调度算法.
E-mail: ylzhang0319@163.com