软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(12):5480−5494 [doi: 10.13328/j.cnki.jos.007432] [CSTR: 32375.14.jos.007432]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

云边协同的深度学习作业调度方法*

谷典典, 金 鑫, 刘譞哲

(北京大学 计算机学院, 北京 100871)

通信作者: 金鑫, E-mail: xinjinpku@pku.edu.cn

摘 要: 边缘服务器 (edge server) 为移动智能应用提供了低延时、高性能的服务. 然而, 由于边缘服务器上的负载量随时间波动较大, 在负载较低的时刻, 许多边缘服务器处于闲置状态, 其计算资源并没有得到充分利用. 与边缘服务器的利用率不同, 随着人工智能技术在人们生活中的应用越来越广泛, 云计算集群中的计算资源对于深度学习训练作业来说仍较为紧张. 现有的集群调度策略不能有效利用云计算集群外的空闲计算资源, 而有效利用云计算集群外的空闲计算资源可以缓解云计算集群的资源紧张问题, 从而使得更多截止期敏感的深度学习训练作业在截止期之前完成. 针对这一问题, 设计一种面向截止期敏感的深度学习训练作业的集群调度策略, 协同调度云计算资源和空闲的边缘计算资源, 充分利用不同深度学习训练作业的性能特征和空闲的边缘服务器设备, 使得更多的截止期敏感的深度学习训练作业在其截止期之前完成. 最后, 实验结果表明, 云边协同的调度方法在提升作业的截止期满足率方面优于其他基线方法, 并有效地利用空闲的边缘服务器设备, 提高计算资源的利用率.

关键词: 云边协同; 深度学习训练; 集群调度; 集群管理; 截止期

中图法分类号: TP311

中文引用格式: 谷典典, 金鑫, 刘譞哲. 云边协同的深度学习作业调度方法. 软件学报, 2025, 36(12): 5480–5494. http://www.jos.org.cn/1000-9825/7432.htm

英文引用格式: Gu DD, Jin X, Liu XZ. Cloud-edge Coordinated Scheduling Method for Deep Learning Jobs. Ruan Jian Xue Bao/Journal of Software, 2025, 36(12): 5480–5494 (in Chinese). http://www.jos.org.cn/1000-9825/7432.htm

Cloud-edge Coordinated Scheduling Method for Deep Learning Jobs

GU Dian-Dian, JIN Xin, LIU Xuan-Zhe

(School of Computer Science, Peking University, Beijing 100871, China)

Abstract: Edge servers provide low-latency, high-performance services for mobile intelligent applications. However, due to significant fluctuations in the load on edge servers over time, many edge servers remain idle during periods of low load, and their computational resources are not fully utilized. In contrast to the underutilization of edge servers, computing resources in cloud computing clusters remain relatively scarce for deep learning training tasks as artificial intelligence becomes more widely applied in daily life. Existing cluster scheduling strategies fail to efficiently utilize idle computing resources outside of cloud computing clusters. Effectively utilizing these idle resources can alleviate the resource constraints in cloud computing clusters, thus enabling more deadline-sensitive deep learning training tasks to be completed before their deadlines. To address this issue, this study proposes a cluster scheduling strategy for deadline-sensitive deep learning training tasks, which coordinates the scheduling of cloud computing resources and idle edge computing resources. This strategy fully leverages the performance characteristics of different deep learning tasks and the availability of idle edge server devices, allowing more deadline-sensitive tasks to be completed on time. Simulation results demonstrate that the cloud-edge collaborative scheduling method outperforms other benchmark methods in improving the deadline satisfaction ratio and effectively utilizes idle edge server devices.

Key words: cloud-edge coordination; deep learning training; cluster scheduling; cluster management; deadline

---

* 基金项目: 国家重点研发计划 (2022YFB4500700); 国家杰出青年科学基金 (62325201); 国家自然科学基金 (62172008)
收稿时间: 2024-01-08; 修改时间: 2024-12-22, 2025-02-19; 采用时间: 2025-03-23; jos 在线出版时间: 2025-07-23
CNKI 网络首发时间: 2025-07-23

随着物联网 (IoT) 设备的普及和数据量的爆炸性增长, 越来越多的边缘设备连接到互联网, 产生大量的数据[1–3]. 传统云计算将数据传输到云端进行处理, 但是由于网络延迟和带宽限制, 这种方式无法满足实时处理和分析的需求. 因此, 边缘计算应运而生: 其本质是将数据处理和分析从云端转移到网络的边缘, 即靠近数据源的地方, 以此减少数据传输的延迟和带宽消耗, 提高数据处理的实时性和效率. 同时, 边缘计算还可以降低对云端服务器的依赖, 提高系统的可靠性和安全性. 为了满足这样的需求, 边缘服务器被部署在靠近数据源或用户端的网络边缘, 用来处理和存储来自物联网设备、移动设备或其他边缘设备的数据.

然而, 在实际部署中, 这些边缘服务器并没有每时每刻都得到充分的利用. 近年对商业边缘平台的大规模实证研究表明[1,4,5], 这一现象的产生是因为边缘服务器所处理的作业负载主要由用户触发, 在某些时间段内, 边缘用户对服务的需求量会突然增加, 导致边缘服务器的负载增加; 而其他时刻边缘服务器处理的负载量较低. 由于边缘服务器的利用具有明显的潮汐现象, 许多边缘服务器的计算资源没有得到充分的利用.

与此同时, 云计算中的深度学习 (deep learning, DL) 训练等场景需要大量的计算资源, 由于计算资源紧缺, 经常出现多个作业在队列中等待资源的现象[6]. 因此, 为了充分利用边缘服务器上没有被利用的资源, 本文提出了一种面向截止期敏感的深度学习训练作业的集群调度策略 EdgeFlow, 协同调度云计算资源和空闲的边缘计算资源. 该调度策略选择合适的深度学习训练作业在边缘服务器上执行, 以缓解云计算集群中计算资源紧张的问题, 并利用弹性训练的优势, 根据集群资源利用率和作业的截止期实时改变云计算集群中的作业使用的 GPU 等计算资源量, 既可以提高作业截止期满足率, 又能提高边缘服务器的利用率.

在这样的云边协同的场景下, 深度学习训练作业的调度问题面临以下两个难点: 首先, 边缘服务器和云计算集群中的服务器、网络带宽等硬件资源具有异构性. 与云计算集群中常用于深度学习训练的 GPU 服务器相比, 边缘服务器的计算能力往往较差, 且边缘服务器之间、边缘服务器和云计算集群之间的网络带宽较小. 因此, 如何根据作业的截止期需求以及云、边缘的资源情况进行合理的资源分配, 具有一定的困难. 已有的面向截止期敏感的深度学习作业调度方法均未考虑云和边缘服务器硬件的差异以及作业在不同硬件上的特性. 其次, 未来到来的深度学习训练作业的时间、作业特性以及不同时刻可用的边缘服务器资源数量对于调度器来说是未知的, 因此调度器不能直接基于所有信息直接做约束求解.

为了解决这样的挑战, 本文定义了云资源使用优先级, 以考虑到不同硬件和作业特点. 在每个事件 (新作业到来、作业结束、可用的边缘服务器数量改变) 发生时, 本文算法 EdgeFlow 会根据作业的优先级和资源使用情况, 对每个作业使用的资源类型和数量进行重新调度, 以对不同的事件作出灵活的实施调整. 该算法的实现可以提高边缘服务器的资源利用率, 同时使得更多的截止期敏感作业可以在截止期之前完成.

本文的主要贡献有: (1) 分析了在面向截止期敏感的深度学习训练作业的云计算集群中, 云边协同计算在提升资源利用率、提高截止期满足率上的优势; (2) 提出了面向深度学习作业的云边协同调度策略 EdgeFlow, 包括一个云计算集群的资源分配方法和一个作业分载算法; (3) 通过实验验证了 EdgeFlow 对于提升截止期满足率的有效性, 并分析了 EdgeFlow 能够提升截止期满足率的原因.

本文第 1 节介绍边缘计算和深度学习作业调度的研究背景和相关工作. 第 2 节介绍云边协同可以为面向深度学习作业的集群调度带来的优势. 第 3 节介绍 EdgeFlow 系统架构及其所解决的问题的描述. 第 4 节介绍本文设计的云边协同的深度学习作业调度方法. 第 5 节通过实验验证了所提方法的有效性. 第 6 节讨论本文的局限性与未来工作. 最后总结全文.

1 研究背景及相关工作

1.1 边缘计算

随着信息技术和通信技术的快速发展, 各种用户设备和物联网设备, 比如智能手机、摄像头、AR/VR 眼镜等的普及, 使得各种实时服务的需求不断增多. 伴随着 4G/5G 以及 WiFi 的发展, 用户设备接入边缘设备 (比如移动

基站、服务器、路由器等) 的延迟得到了显著降低. 而随着用户设备算力的不断提升, 仅使用边缘设备完成计算成为可能. 因此在这样的背景下, 将计算资源从云计算集群中的服务器下发到靠近用户的边缘设备成为了实现大规模实时计算的必然趋势. 在这种新的计算模式下, 数据和服务信息不必经过主干网络, 使得服务的实时性方面和数据的隐私安全性都得到了提升.

与云计算集群中的服务器相比, 边缘服务器通常价格低廉, 用于处理非计算密集型的负载. 因此和云计算集群中的服务器相比, 边缘服务器通常不会使用最新型的服务器, 具有较少的算力. 另外, 边缘服务器和云计算集群中的服务器之间的网络带宽通常较小, 边缘服务器和云计算集群之间数据传输通常具有较大的延迟. 和云计算服务器相比, 边缘服务器在地理上的分布更为分散[5], 边缘服务器之间的通信也同样具有较大的延迟.

由于边缘服务器的作业负载主要由用户触发产生, 而用户对很多服务的访问频率通常有较明显的高峰期和低谷期, 因此, 边缘服务器的利用率存在显著的潮汐现象. 例如, 有研究表明, 午夜时边缘片上系统 (system-on-chip, SoC) 的 CPU 使用率约为高峰时段的 1/50, 且在大约一半的时间里边缘设备的利用率低于 50%[5]; 阿里巴巴的边缘服务供应平台 ENS[4] 的资源利用率较低, 且资源使用量具有波动性. 由于边缘服务器上访问量低谷期的时间较长, 边缘服务器资源在较长的时间里都被闲置, 无法得到充分利用, 造成计算资源浪费.

目前已有工作利用空闲的边缘服务器执行其他作业负载, Xu 等人[5]利用边缘 SoC 设备的空闲时段进行分布式机器学习训练. 然而该方法只适用于部署单个深度学习训练作业, 若有多个深度学习训练作业可被部署到边缘设备, 该方法无法对在何时部署何作业进行判断; 且该方法只适用于特定的 SoC 设备, 无法扩展至使用 GPU 等硬件训练深度学习模型的情况.

1.2 深度学习作业调度

随着深度学习的繁荣发展, 其在键盘输入法和聊天软件中常见的语音文本转换[7]、能够提高通行效率的自动驾驶[8]等多个领域都有广泛的应用. 如今, 深度学习模型的训练变得越来越耗时且资源密集, 专用的 GPU 服务器被集成到 GPU 集群中, 以支持这些计算密集型的作业负载. 在这种情况下, 为了降低运营成本并提高资源利用率, 为 GPU 集群中的深度学习训练作业设计一个高效的调度器变得至关重要.

然而, 传统的针对大数据或高性能计算 (high performance computing, HPC) 作业负载设计的方法没有考虑深度学习训练作业的特有特征, 不能充分支持深度学习作业负载, 以充分利用 GPU 资源. 因此, 近期许多研究提出了专门为 GPU 集群中的深度学习负载量身定制的调度器.

面向深度学习作业的 GPU 调度器具有多种不同的调度目标, 已有研究分别针对不同的调度目标进行了面向深度学习作业的调度算法设计. 针对优化作业完成时间的调度目标, Gu 等人[9]通过高效调度并且合适地放置深度学习作业, 设计了 Tiresias 调度算法, 减少深度学习训练作业的作业完成时间 (job completion time, JCT). Hwang 等人[10]利用弹性训练, 在作业执行的不同时刻根据作业本身的特点以及集群资源的紧张程度动态改变其使用的 GPU 数量, 进一步对集群中所有作业的平均完成时间进行优化. 针对提高资源利用率的调度目标, Xiao 等人[11]针对深度学习作业的特点, 让作业具有迁移能力以减小其对 GPU 的亲和性, 以及模型之间的相互作用对作业的影响, 设计了为面向深度学习作业的 GPU 集群调度器 Gandiva, 从而大幅度提升集群的利用率. 针对提高公平性的调度目标, Mahajan 等人[6]提出了完成时间公平性 (finish-time-fairness) 的 GPU 资源调度指标的定义, 并根据这一评价指标设计了调度算法 Themis.

除了这些常见的指标以外, 深度学习训练的开发者同样对作业完成时间有各自的预期, 例如, 一些服务的开发商希望推荐模型的训练作业可以在产品上线之前完成[12], 我们称这样的作业预期完成时间为截止期. 针对开发者的这一需求, Gao 等人[12]和 Gu 等人[13]都分别根据开发者对作业完成时间的不同预期水平, 将提交到云计算集群的不同深度学习训练作业分类, 并设计算法尽量使得所提交作业的截止期之前完成.

然而, 在云边协同执行深度学习训练作业的场景下, 一方面, 边缘服务器的性能和云计算集群中服务器的算力差距较大; 另一方面, 边-边、云-边、云-云设备之间网络性能同样有较大的差异. 已有的研究工作忽视边缘服务器和云计算集群的硬件特征, 因此如果直接应用到云边协同场景下, 不能得到较好的调度效果. 为了弥补现有研究的

不足, 本文在充分利用空闲的边缘服务器资源的同时, 综合考虑不同服务器以及服务器之间网络带宽的特点, 随时灵活调整每一个作业使用的资源数量和类型, 以使得更多的深度学习训练作业可以在截止期之前完成.

2 云边协同的优势

尽管边缘服务器的性能往往不如云计算集群中的 GPU 服务器, 且将作业从云计算集群迁移到边缘服务器具有不可忽略的时间开销, 但将部分作业负载迁移到边缘服务器可以为保留在云计算集群中的作业留下更充足的计算资源. 与此同时, 保留在云计算集群中的作业可以在训练的过程中不再严格使用开发者指定的 GPU 数量执行训练作业, 而是在不同时刻动态改变其使用的 GPU 数量, 从而加速作业的执行, 使得所有开发者提交的深度学习训练作业的截止期满足率有所提升. 我们称使用 GPU 数量动态改变的训练过程为弹性训练.

为了验证使用空闲边缘服务器协助云计算集群可能显著提升截止期满足率, 本文首先从阿里巴巴公开的向其 GPU 生产集群提交的作业历史[14] 中选取 10 个作业. 原始作业提交历史中包含每个作业的提交时间、使用 GPU 数量和作业持续时间等信息, 但缺少作业训练的模型、批大小 (batch size)、截止期等重要信息. 本文将在第 5.1 节详细介绍如何利用已有信息生成实验所需要的完整信息.

然后, 本文对只有云计算集群的环境以及云边协同环境下的执行过程分别进行模拟. 所模拟的云计算集群环境包含一个含有 8 个 NVIDIA A100 GPU 的云端服务器, 云边协同环境在该云计算集群所拥有的计算资源的基础上, 还包含一个算力比 NVIDIA A100 GPU 更低的 NVIDIA V100 GPU 的边缘服务器. 本文将在第 5.1 节介绍模拟器的实现.

模拟结果显示, 在只有云计算集群的环境中, 7 个作业可以在截止期之前完成. 而在仅利用了一个边缘服务器, 并将其中一个作业迁移至边缘服务器之后, 可以在截止期之前完成的作业数量增加至 9 个. 因此, 即使是少量的边缘服务器也可能提升深度学习训练作业的截止期满足率. 基于这样的发现, 本文设计了云边协同的深度学习作业调度方法 EdgeFlow.

3 系统架构及问题描述

本节的场景介绍包括系统架构以及作业在云计算集群和边缘服务器的执行模型, 然后对云边协同场景下的深度学习作业调度的优化问题做出描述.

3.1 系统架构

图 1 展示了本文设计的云边协同的深度学习作业调度方法 EdgeFlow 的系统架构图. 首先, 深度学习开发者提交深度学习训练作业. 在执行每个作业之前, 作业分析器会使用一定的云计算集群资源预执行作业, 以分析其在使用不同数量的云计算资源时的吞吐率 (单位时间内可以执行的批数量) 表现. EdgeFlow 的资源分配模块通过执行集群监测模块实时获取云计算集群中资源的使用情况以及可用的边缘服务器数量, 并使用基于优先级的资源分配算法决定每一个作业将被分载至边缘服务器还是保留在云计算集群, 如果作业被保留在云计算集群, 则该模块还将为每一个提交到云计算平台的作业分配其使用的 GPU 资源数量. 在每次调度事件 (如新作业提交、作业完成、可用边缘服务器数量变化等) 时, 资源分配模块可能会通过弹性缩放 (即根据已有作业的截止期和可用 GPU 的数量调整作业被分配的 GPU 数量) 更新某些作业的资源分配, 即改变其使用的云计算集群中的 GPU 数量, 或改变其使用的计算资源类型 (云计算集群中的 GPU 或边缘服务器). 如果作业被分配的资源发生改变, 为保证每个作业的收敛性不受影响, 该模块会保证作业的全局批大小 (global batch size) 保持不变, 并计算在使用新的 GPU 个数的情况下每个作业在每块 GPU 上的本地批大小 (local batch size). 作业放置模块根据云计算集群以及边缘服务器之间网络带宽的拓扑结构, 选择每个作业具体放置的 GPU 设备. 随后, 作业放置模块将作业发送到弹性训练执行器. 弹性训练执行器使得深度学习作业在改变执行所用的 GPU 数量时可以维持原有训练进度并继续按照正确的参数训练. 该模块是可以被替换的, 任何支持弹性训练的深度学习训练框架都可以满足 EdgeFlow 的弹性训练需求.

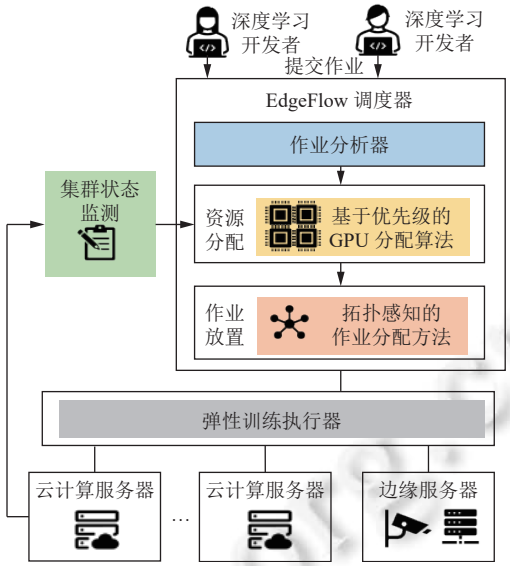


图 1　云边协同的深度学习作业调度方法 EdgeFlow 系统架构图

3.2　作业执行模型

为了让更多作业在截止期之前完成, EdgeFlow 借助弹性训练[10,13,15], 根据开发者所提交的作业的截止期和集群资源紧张程度实时调整每个作业执行时用的 GPU 数量或使用的资源类型. 因此, 每个作业所使用的 GPU 数量和类型在其生命周期中是动态变化的. 本文使用 $S_i=\{s_{1,i},s_{2,i},\ldots,s_{u_i,i}\}$ 表示作业 i 在其生命周期中不同阶段使用的资源类型. 其中 u_i 表示作业 i 生命周期中经历的事件 (新作业到来、作业结束、可用的边缘服务器数量改变) 个数, 作业 i 在其生命周期经历的第 j 个事件之后的一段时间内使用的资源类型 $s_{j,i}\in\{0,1\}$. $s_{j,i}=0$ 表示作业 i 在第 j 个事件之后的一段时间内使用云计算集群中的计算资源, $s_{j,i}=1$ 表示作业 i 在这段时间内只用边缘服务器上的计算资源. 由于云计算集群和边缘服务器之间的网络带宽较小, 为了高效执行作业, EdgeFlow 只会为作业分配云计算集群中的 GPU、边缘服务器两种计算资源其中的一种.

本文使用 $A_i=\{a_{1,i},a_{2,i},\ldots,a_{u_i,i}\}$ 表示作业 i 的资源分配方案, 即在作业 i 生命周期中不同阶段使用的资源数量. 其中, $a_{j,i}$ 表示作业 i 在第 j 个事件之后的一段时间内使用 $a_{j,i}$ 个单位 (GPU 个数) 的资源进行数据并行分布式训练, $a_{j,i}$ 为正整数. 边缘服务器上可能只有一个 GPU, 也可能有多个 GPU, 这些 GPU 未必同时处于空闲状态. 边缘服务器之间的通信带宽较小, 导致使用边缘服务器进行跨机分布式训练的通信开销过大. 因此, EdgeFlow 为了让边缘服务器上的作业高效执行, 同时统一考虑不同边缘服务器的配置和资源使用情况, 本文较为保守地使用边缘服务器进行单 GPU 训练、使用云计算集群中的 GPU 进行分布式训练或单 GPU 训练. 因此, 若 $s_{j,i}=1$, 则 $a_{j,i}=1$; 若 $s_{j,i}=0$, $a_{j,i}\geqslant 1$. 若 $s_{j,i}=0$, $a_{j,i}$ 的取值仍需要满足一些其他的条件: 一方面, 深度学习训练作业使用的资源单位个数较少时, 为保持全局批大小不变, 本地批大小较大; 为避免作业的执行过程中需要每个单位的计算资源提供的内存小于其内存上限, 深度学习训练作业所使用的资源单位个数不能过小. 另一方面, 在深度学习训练作业使用的资源单位个数不超过某个数值时, 其使用的资源数量越多, 训练的吞吐率越大; 但在其使用的资源数量超过该数值之后, 由于多 GPU 和多服务器之间协作进行分布式训练的通信开销较大, 作业的吞吐率反而会下降, 因此深度学习训练作业所使用的资源单位个数不能过大. 总结来说, $a_{j,i}$ 的取值范围为 $\min(a_i)\leqslant a_{j,i}\leqslant\max(a_i)$, 其中 $\min(a_i)$ 为作业 i 训练时不会发生内存溢出错误时使用的资源数量最小值, $\max(a_i)$ 为作业 i 的吞吐率随着使用的 GPU 数量增加而增加时使用的最多的 GPU 数量.

深度学习训练作业往往具有终止条件, 例如: 累积训练的批 (*iteration*) 数量不超过某个最大值、损失值 (*loss*) 不超过某个指定的数值等. 为了防止因模型训练不收敛导致训练作业永远不会停止, 对于作业 i, 深度学习开发者

往往会设定一个累积训练的批数量的上限值 $iterations_i$. 即使作业 i 具有损失值不超过某个指定的数值等其他终止条件, 作业 i 的累积训练的批数量也不会超过 $iterations_i$.

综上, 若共有 n 个深度学习训练作业被提交至调度器, 则深度学习训练作业 i 需要满足的条件为:

$$\begin{cases} \forall 1 \leqslant j \leqslant u_i,\ \min(a_i) \leqslant a_{j,i} \leqslant \max(a_i), & \text{if } s_{j,i}=0 \\ \forall 1 \leqslant j \leqslant u_i,\ a_{j,i}=1, & \text{if } s_{j,i}=1 \end{cases} \tag{1}$$

$$\sum_{j=1}^{u_i} t_{j,i} \times ((1-s_{j,i}) \times tpt_{i,a_{j,i}}^{\text{cloud}} + s_{j,i} \times tpt_i^{\text{edge}}) \geqslant iterations_i \tag{2}$$

其中, u_i 表示作业 i 其生命周期中除了“作业 i 完成”以外发生的所有事件的个数, $tpt_{i,a_{j,i}}^{\text{cloud}}$ 表示作业 i 在使用 $a_{j,i}$ 个云计算集群中的 GPU 时的吞吐率 (单位时间可以执行的批数量), tpt_i^{edge} 表示作业 i 在单个边缘服务器上执行时的吞吐率. 公式 (1) 为作业 i 使用的资源数量限制, 公式 (2) 为作业 i 在其生命周期内累积完成的批数量满足终止条件. 在满足公式 (1) 的条件下, 作业使用的 GPU 数量越多, 吞吐率越大. 因此, 用 x 和 y 表示作业 a_i 可以使用的 GPU 数量, 如果 $\min(a_i) \leqslant x, y \leqslant \max(a_i)$ 且 $x<y$, 则 $tpt_{i,x}^{\text{cloud}} < tpt_{i,y}^{\text{cloud}}$; 由于边缘服务器的性能往往不如云计算集群中的 GPU 服务器, 因此有: $tpt_i^{\text{edge}} < tpt_{i,1}^{\text{cloud}}$.

在满足公式 (1) 和公式 (2) 的条件下, 若作业 i 满足:

$$t_{u_i,i} \leqslant ddl_i \tag{3}$$

则作业 i 在其截止期之前完成. 其中, $t_{u_i,i}$ 表示作业 i 的最后一个事件 (u_i, 即作业 i 完成) 发生的时间.

3.3 问题描述

综合以上设定, 本文希望解决的优化问题为:

$$\max \frac{\left|\left\{i \middle| t_{u_i,i} \leqslant ddl_i\right\}\right|}{n} \tag{4}$$

$$\begin{cases} \min(a_i) \leqslant a_{j,i} \leqslant \max(a_i), & \text{if } s_{j,i}=0 \\ a_{j,i}=1, & \text{if } s_{j,i}=1 \\ \text{s.t. } \forall 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant u_i & \end{cases} \tag{5}$$

$$\forall 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant u_i,\ \sum_{j=1}^{u_i} t_{j,i} \times ((1-s_{j,i}) \times tpt_{i,a_{j,i}}^{\text{cloud}} + s_{j,i} \times tpt_i^{\text{edge}}) \geqslant iterations_i \tag{6}$$

其中, 公式 (4) 表示优化目标为最大化在截止期之前完成的作业数量, 公式 (5) 和公式 (6) 表示每个作业都需满足公式 (1) 和公式 (2) 的条件.

4 云边协同的深度学习作业调度方法 EdgeFlow

4.1 作业分析器

为了准确判断作业采用不同资源分配方案时能否在截止期之前完成, EdgeFlow 执行每一个作业之前需要先通过作业分析器短暂地预执行作业. 深度学习作业往往不断重复执行大量的迭代过程, 在其稳定训练的时候, 每一次迭代所需时间几乎一致, 整体作业的吞吐率也较为稳定. 因此, 对于每一种资源配置, 执行少量的几个训练批就能得到在相应配置下稳定的训练吞吐率. 根据深度学习训练作业的这一特点, EdgeFlow 作业分析器使用云计算集群的 GPU 资源, 对每一个作业在使用不同数量的云计算集群资源时的吞吐率进行分析, 分析得到的吞吐率数据将被 EdgeFlow 的资源分配模块用来模拟作业在不同配置下的执行进度. 预执行作业会带来不可避免的时间开销, 图 2 展示了 64 个 NVIDIA A100 GPU 上通过作业分析器预执行作业所需时间的平均值, 模型训练的配置如表 1 所示. 和大多数深度学习训练作业所需的数小时甚至数天的执行时间相比, 这样的时间开销对作业整体执行时间的影响较小[12,13].

可用的边缘服务器资源数量是动态变化的, 在边缘服务器原本需要服务的作业负载量较高时, 可能没有深度学习训练作业可用的空闲边缘服务器. 因此, EdgeFlow 不对深度学习训练作业在边缘服务器上的吞吐率进行分析. 如第 3.2 节提到的, 由于边缘服务器之间的通信带宽较小, 导致使用边缘服务器进行分布式训练的通信开销过大,

因此本文用 $tpt_i^{edge}=\lambda_i\times tpt_{i,1}^{cloud}$ 来预估作业 i 使用一个边缘服务器时的吞吐率, 其中 $tpt_{i,1}^{cloud}$ 为作业 i 使用一个云计算集群中的 GPU 时的吞吐率, λ_i 是一个参数, $\lambda_i\in(0, 1)$, 其取值取决于边缘服务器和云计算集群中 GPU 的性能以及具体的深度学习训练作业的特征.

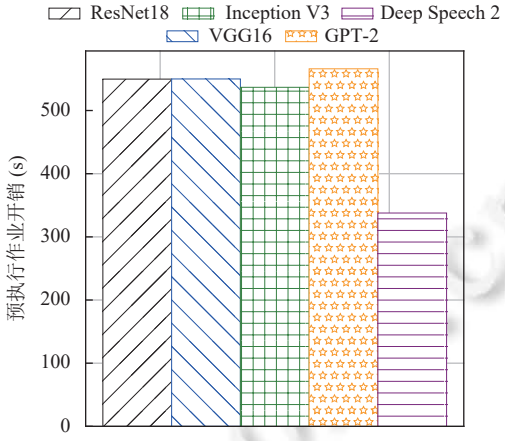


图 2 作业分析器预执行作业的时间开销

表 1 实验评估中用到的深度学习模型及其训练设定

训练作业类型	数据集	深度学习模型	批大小
计算机视觉	ImageNet[16]	ResNet18[17]	64, 128, 256
		VGG16[18]	64, 128, 256
		Inception V3[19]	32, 64, 128
自然语言处理	aclImdb V1[20]	GPT-2[21]	16, 32, 64
语音识别	LibriSpeech[22]	Deep Speech 2[23]	16, 32

为了尽量准确地分析作业执行进度, EdgeFlow 会在每个作业在云计算集群或边缘服务器上实际执行的过程中, 对该配置下的吞吐率进行记录, 修正预执行或预估得到的吞吐率数据, 同时也修正不同作业的 λ_i 参数. 根据我们在 NVIDIA A100 GPU 和 NVIDIA V100 GPU 上执行作业分析的测试, 在作业执行期间的分析记录的开销约为每次 0.01 s, 与深度学习训练作业的执行时间 (数小时甚至数天) 相比, 这样的时间开销可忽略不计.

4.2 云边协同的深度学习作业资源分配步骤

对于调度器来说, 未来的深度学习训练作业的时间、作业特性以及不同时刻可用的边缘服务器资源数量是未知的, 因此调度器不能基于所有信息直接做离线约束求解. 因此, EdgeFlow 的资源分配模块使用在线的资源分配算法, 在每个事件 (新作业到来、作业结束、可用的边缘服务器数量改变) 发生时对尚未结束的所有作业重新分配资源. 在每一次需要重新分配资源的时候, EdgeFlow 的调度分为 3 个步骤.

首先, EdgeFlow 考虑云计算集群中的资源, 参考已有工作的方法, 计算每个作业的最小满足份额, 并为每个作业根据最小满足份额进行资源分配[13]. 这样的方法可以使得能在截止期之前完成的作业使用最少量的计算资源, 从而把资源分配给尽可能多的作业, 让尽可能多的作业在截止期之前完成. 如果有的作业因为集群资源紧张或期截止期过于紧张而不能在截止期之前完成, EdgeFlow 尽量为其分配最多的可用计算资源, 使其尽早完成并释放 GPU 资源, 从而为未来提交的作业预留云计算集群中的 GPU.

随后, 如果此刻有可用的空闲边缘服务器, EdgeFlow 会选取部分作业, 将其迁移并分载至边缘服务器执行. 这一过程中涉及云计算资源和边缘服务器资源的协同调度, 本文将在第 4.3 节介绍 EdgeFlow 所使用的云边协同的深度学习作业分载算法.

最后, 如果云计算集群中仍有没有被利用的 GPU 资源, 在不会让已有作业的训练速度变慢的情况下, EdgeFlow 进一步将这些 GPU 资源分配给云计算集群中的作业, 加速这些作业的执行, 从而使系统能够容纳更多的未来到来的作业. 在这一过程中, EdgeFlow 采用已有工作[13]的资源分配算法, 将剩余的 GPU 分配给 GPU 资源利用率更高 (即新增加一块 GPU 后, 节省的作业完成时间最长) 的作业, 从而高效利用这些 GPU 资源.

在每次调度事件发生并进行再调度时, 状态如何保存取决于弹性训练执行器的实现. 一种实现方式是将被调度的作业的状态 (即深度学习模型的权重数据、优化器状态等) 以 checkpoint 文件格式或其他格式保存至云计算集群存储, 或云计算集群和边缘服务器均可访问的远程存储. 这种作业迁移方法的开销较大. 若作业不在云计算集群和边缘服务器之间迁移, 仅在云计算集群内部迁移, 则弹性训练执行器可以将作业状态保存至内存. 这种作业迁移方法的开销较小. 如果使用第 1 种状态保存方式, 被迁移作业的目标服务器上的训练进程可以从云计算集群存储或远程存储中读取作业状态, 将之前训练的中间状态加载至训练进程中, 并从该状态开始继续训练. 如果采用第 2 种状态保存方式, 被迁移作业的目标服务器上的训练进程可以和原服务器建立通信连接, 从原服务器接收作业状态数据. 本文将在第 5.3 节分析作业迁移的时间开销.

4.3 云边协同的深度学习作业分载算法

由于不同的作业在不同硬件上的吞吐率表现差异较大, EdgeFlow 设计分载算法, 在每个作业根据最小满足份额得到了初步资源分配方案后, 选取部分作业, 将其迁移至边缘服务器执行. 这一过程中对作业的选取遵循 3 个规则.

(1) 在云计算集群和边缘服务器之间进行作业迁移具有较大的时间开销, 为了避免对同一作业进行反复迁移并减少迁移次数, EdgeFlow 优先让已经被分载到边缘服务器的作业保留在边缘服务器上执行.

(2) 云计算集群中有的作业即使在边缘服务器上以较慢的速度执行, 也可以在截止期之前完成. 如果在执行规则 (1) 后, 仍有空闲的边缘服务器, 那么 EdgeFlow 会将这样的作业迁移至边缘服务器上执行. 这样既能保证这些作业满足截止期需求, 也能为其他作业留出云计算集群资源, 从而加速留在云计算集群中的作业的执行, 部分原本因集群资源紧张而无法在截止期之前完成的作业可能因此满足了其截止期需求. 我们将这些作业的集合记为 J_{edge}. 然而, 如果在将 J_{edge} 中的作业分载之前, 云计算集群中仍有空闲的 GPU 资源, J_{edge} 中的作业不会被分载至边缘服务器. 根据第 4.2 节, 资源分配步骤中的第 1 步完成后云计算集群中仍有空闲的 GPU 意味着即使为不满足截止期的作业再增加 GPU 也无法提升吞吐率, 作业分载节省下的 GPU 同样不会使更多的作业满足截止期. 因此, 是否将 J_{edge} 中的作业分载至边缘服务器取决于云计算集群中是否有剩余计算资源: 如果有剩余的计算资源, 将 J_{edge} 中的作业保留在云计算集群中不仅能保证这些作业满足截止期需求, 还能使得这些作业更早完成. 值得注意的是, 只有内存需求不超过边缘服务器可提供的内存的作业可以被分载至边缘服务器.

(3) 如果在执行了规则 (2) 之后, 仍有剩余的边缘服务器, EdgeFlow 仍可能选取部分作业迁移至边缘服务器. 这是因为, 对于一些作业来说, 即使进一步增加其使用的计算资源数量, 其完成时间不会有较大的改变, 仍无法满足截止期需求; 而对于另一些作业来说, 如果进一步增加其使用的计算资源数量, 其完成时间可能明显更接近截止期, 甚至可能满足截止期需求. 因此, 如果将很难满足截止期需求的作业迁移至边缘服务器, 并用曾经分配给这些作业的资源加速留在云计算集群中的作业, 可能进一步提升截止期满足率. 本文根据不同作业特点, 定义作业“云资源使用优先级”为当下时刻为作业 i 多分配一个 GPU 之后其完成时间与截止期的距离:

$$priority_i = ddl_i - end\_time(i, \tilde{A}_i) \tag{7}$$

其中, $\tilde{A}_i = \{a_{1,i}+1, a_{2,i}, \ldots, a_{\tilde{u}_i,i}\}$ 表示作业 i 从此刻开始到下一个事件到来之前, 比原资源分配方案多使用一个 GPU, 因为这一段时间内作业 i 的吞吐率增加, 其完成时间会提前, 因此其生命周期中经历的事件数量可能有所变化, 本文将作业 i 在第 1 个事件到来前多使用一个 GPU 后其生命周期中经历的事件个数记为 $\tilde{u}_i$. $end\_time(i, \tilde{A}_i)$ 为作业 i 不再使用的资源分配方案 A_i, 而使用资源分配方案 $\tilde{A}_i$ 之后的预计完成时间. 云资源使用优先级的计算过程如算法 1 所示. $priority_i \geqslant 0$ 代表作业 i 使用资源分配方案 $\tilde{A}_i$ 后可以满足截止期需求; $priority_i < 0$ 代表作业 i 使用资源分配方案 $\tilde{A}_i$ 后仍然不能满足截止期需求; $priority_i > priority_j$ 代表作业 i 多使用一个 GPU 后的完成时间比作业 j 多使用一个 GPU 后的完成时间更可能满足截止期需求.

算法 1. 云边协同的深度学习作业分载算法.

输入: 当前时间 *current_time*, 当前尚未结束的所有作业的集合 *Job*, 当前初步资源分配方案下是否存在作业无法满足截止期要求 *violated*, 当前云计算集群中是否含有空闲计算资源 *has_free_gpu*, 云计算集群含有的 GPU 个数 *G*, 可用边缘服务器数量 *E*, 每个边缘服务器可用的最大内存 *edge_memory*;
输出: 分载到边缘的作业集合 J_{edge}.

```
1. J_edge ← {} //初始化 J_edge
2. Q ← {} //初始化优先队列 Q
3. //执行规则 (1)
4. for i in Job do
5.    if E>0 then
6.       if i.already_on_edge is True then
7.          J_edge.append(i) //迁移 Job 至边缘服务器
8.          E ← E–1
9. //执行规则 (2)
10. if has_free_gpu then
11.    for i in Job do
12.       if !i.already_on_edge then
13.          end_time_on_edge ← current_time+i.iterations_left/i.tpt_edge
14.          if E > 0 and i.memory < edge_memory and end_time_on_edge≤i.ddl then
15.             J_edge.append(i) //迁移 Job 至边缘服务器
16.             E ← E–1
17. //执行规则 (3)
18. if violated then
19.    for i in Job do
20.       if !i.already_on_edge and get_end_time(i, A_i)>i.ddl and A_i [current_time]<i.max_gpu then
21.          i.priority ← i.ddl–get_end_time(i, Ã_i)
22.          Q.insert(i)
23.    while E>0 and !Q.empty() do
24.       i ← Q.dequeue()
25.       J_edge.append(i) //迁移 Job 至边缘服务器
26.       E ← E–1
```

EdgeFlow 根据云资源使用优先级对仍在云计算集群中的作业进行排序, 并将云资源使用优先级更高的作业保留在云计算集群中. 值得注意的是, 在规则 (3) 中, 如果一个作业使用现有的云计算集群资源可以在截止期之前完成, EdgeFlow 不会将其迁移至边缘服务器, 以避免其截止期需求不再被满足. 与规则 (2) 相同, 只有内存需求不超过边缘服务器可提供的内存的作业可以被分载至边缘服务器.

值得注意的是, 如果将无论如何都无法满足截止期的作业分配到边缘服务器, 会进一步增加这些作业的时延. 然而, 本文的优化目标是提高作业的截止期满足率. 因此, 如果一个作业无论如何都无法满足截止期, 那么即使避免了这一时延, 其截止期也无法被满足. 如果将这些作业分配到边缘服务器, 不仅不会影响其他作业的截止期满足, 还会为其他可能在截止期之前完成的作业预留云计算集群中的 GPU 资源. 这对于本文的优化目标来说是一个更有利的选择.

4.4 作业放置

集群中 GPU 的组织的拓扑结构可以用树状结构来表示[24], 其中 GPU 通过不同类型的链接以不同的带宽连接. 树的叶子节点表示 GPU 设备, 每个内部节点表示特定的链路类型. 图 3 是一个具有 4 层树状结构的服务器的示例. 该服务器有两个通过 QPI 连接的 CPU 插槽; 每个 CPU 通过 PCIe 连接到 4 个 GPU; 同一机架中的多个此类服务器通过机架顶部的交换机 (ToR) 连接.

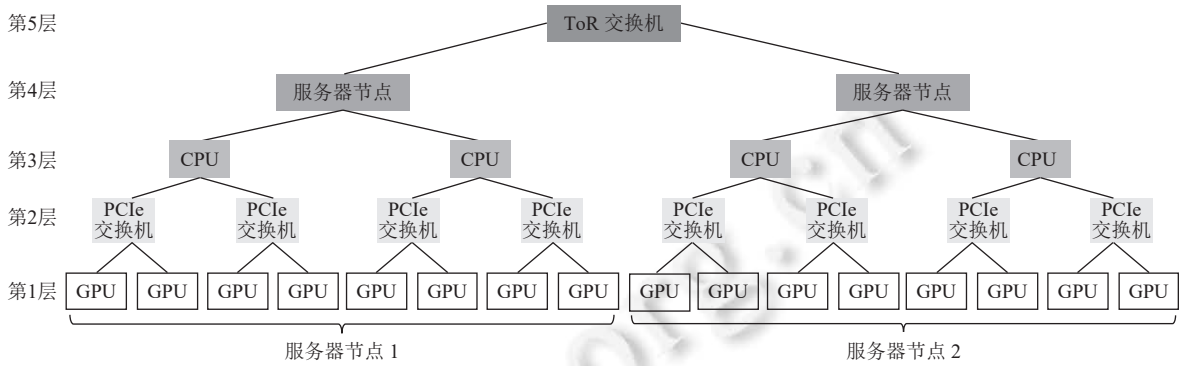


图 3 集群中 GPU 设备的多层树状结构组织

在这样树状组织的拓扑结构下, 由于多层树状结构中每一层之间的链路带宽不同, 每个作业使用的 GPU 设备之间的拓扑关系十分重要. 例如, 使用 4 块 GPU 训练 ResNet50 模型时, 如果 4 块 GPU 分布在 4 个不同的服务器节点上, 其吞吐率不足使用同一个服务器节点上的 4 块 GPU 时的 1/2[11]. 如第 3.2 节提到的, EdgeFlow 为一个作业最多分配一个边缘服务器, 因此本文只考虑在云计算集群中使用多个 GPU 时的作业放置问题.

为了让云计算集群中运行的作业能够充分利用 GPU 之间的最大带宽进行高效通信, EdgeFlow 会在集群空闲的 GPU 中找到空闲 GPU 数量最接近所需 GPU 数量的子树. 同时, EdgeFlow 将每个作业使用的 GPU 数量限制为 2 的幂. 在这样的限制条件下, 如果结合作业迁移技术 (即在作业运行过程中暂停作业, 将其迁移至另外的一组数量相同的 GPU 上继续执行), 可以保证: 如果集群中如果空闲 GPU 数量大于作业所需 GPU 数量, 则一定存在一种作业分配方法, 使得每个作业都使用最大的带宽进行通信[10].

5 实验分析

为了验证 EdgeFlow 方法的有效性, 我们进行了大量的模拟实验: 首先, 将我们提出的方法与其他基线方法在截止期满足率指标进行了实验对比; 然后, 对 EdgeFlow 方法能够提升截止期满足率的因素进行了分析; 最后, 对作业提交间隔、云计算集群大小和可使用的边缘服务器数量等因素对 EdgeFlow 方法效果的影响进行了分析.

5.1 实验设定

为了验证 EdgeFlow 的集群调度效果, 本文首先从阿里巴巴公开的向其 GPU 生产集群提交的作业历史[14]中选取了 100 个作业提交记录 (以下称为“踪迹 1”). 原始提交记录中的每个作业都包含提交时间、使用的 GPU 数量和持续时间的信息, 不包含训练的模型、批大小和截止期信息. 由于 EdgeFlow 的调度方法中每个作业使用的 GPU 数量都是灵活变化的, 因此 EdgeFlow 不会使用原始提交记录中作业使用 GPU 数量这一信息. 对于每个作业, 我们从表 1 中随机选择一个具有代表性的设置, 作为该作业训练的模型和训练批大小. 为了验证 EdgeFlow 在不同生产集群中都具有提高作业截止期满足率的效果, 除了阿里巴巴公开的作业提交历史以外, 我们还选择了微软 Philly 集群的公开提交历史[25]对其效果进行验证 (以下称为“踪迹 2”). 我们参考已有工作的方法[13], 根据作业的原始持续时间为每个作业生成其截止期. 值得注意的是, 即使作业的截止期小于其原本的持续时间, 如果使用更多数量的 GPU, 作业也可能在截止期之前完成.

在实际集群中对 EdgeFlow 以及其基线方法进行实验验证的成本较高. 例如, 在 AWS 平台使用 8 台 p4d.24xlarge

服务器、在踪迹 1 上验证 EdgeFlow 和第 5.2 节中的基线方法的成本约为 7080 美元[26], 执行踪迹 2 需要更多的时长, 成本更高. 鉴于预算限制, 本文开发了一个调度事件模拟器, 通过模拟实验验证 EdgeFlow 的效果. 由于深度学习训练的过程通常是稳定重复迭代的, 模拟迭代的过程可以较好地还原真实训练的过程. 这一方法也是学术界评估面向深度学习作业的调度器的常用方法[9−13]. 因此, 为了在多种规模和云-边协同场景下评估 EdgeFlow, 我们使用具有不同数量的 GPU 的 NVIDIA A100 GPU 和 NVIDIA V100 GPU 分别在表 1 中不同模型设定下执行训练作业, 并记录这些训练设定下的训练作业使用不同型号的 GPU、使用不同数量的 GPU 时的吞吐率数据. 利用这些实际执行的吞吐率数据, 我们建立模拟器模拟不同作业使用不同 GPU 情况下的训练过程. 具体来说, 我们用作业在 NVIDIA A100 GPU 上训练的吞吐率模拟作业在云计算集群中的训练过程, 用作业在 NVIDIA V100 GPU 上训练的吞吐率模拟作业在边缘服务器上的训练过程. 已有研究工作[27]同样使用 NVIDIA V100 GPU 作为边缘设备上的算力. 与此同时, 由于边缘节点的 GPU 算力通常比云计算集群中的高性能算力低, 而 V100 GPU 的算力比 A100 更低, 因此本文选取 V100 GPU 对边缘节点情况进行模拟, 是符合实际场景的. 模拟器会模拟所有作业级别的事件, 包括新作业到来、作业结束、可用的边缘服务器数量改变. 每当事件发生时, 模拟器会计算当前时刻和上一事件发生时的时间差. 两个事件发生之间的时间里, 作业 i 在经历第 j 个事件到第 j+1 个事件这段时间之间完成训练的批个数被模拟为 $t_{j,i}\times((1-s_{j,i})\times tpt_{i,a_{j,i}}^{\text{cloud}}+s_{j,i}\times tpt_i^{\text{edge}}$.

为了使模拟结果更加接近真实情况, 我们还模拟了作业从云计算集群迁移到边缘服务器、在云计算集群内迁移、和在云计算集群内缩放 (作业使用 GPU 的数量变化) 的时间开销. 根据已有工作汇报的结果, 较为先进的弹性训练执行器在同一集群内部的迁移开销约为 1–5 s[13]. 云计算集群和边缘服务器之间通信时延可达数百毫秒至数秒, 因此, 本文较为保守地在模拟云-边作业迁移事件时, 在云计算集群内部作业迁移开销的基础上额外增加 2 s 的时间延时. 本文根据模型权重的实际大小和 5 GB/s 的带宽模拟边缘服务器上的数据传输时间. 模拟器根据作业使用的 GPU 数量以及事件类型, 在每个调度事件中为每个作业模拟相应的时间开销.

5.2 评价指标及基线模型

在本文中, 我们采用评价指标截止期满足率来评估不同调度方法对具有截止期的深度学习训练作业的调度效果. 我们将所提方法 EdgeFlow 与以下集群调度方法进行比较.

(1) Gandiva[11]: Gandiva 是一个面向深度学习作业的 GPU 集群调度器, 它通过动态调整作业的优先级提升集群使用的效率. Gandiva 不是截止期感知的, 且 Gandiva 所调度的作业不是弹性的.

(2) Tiresias[9]: Tiresias 是一个面向深度学习作业的 GPU 集群调度器, 同时考虑作业在时间、空间两个维度上的资源利用, 以减少集群中作业的平均作业完成时间. Tiresias 不是截止期感知的, 且 Tiresias 所调度的作业不是弹性的.

(3) AFS[10]: AFS 面向深度学习作业调度 GPU 集群中的 GPU 资源, 通过平衡资源效率和短作业优先级加速深度学习训练作业的执行. AFS 所调度的作业是弹性的, 但不是截止期感知的.

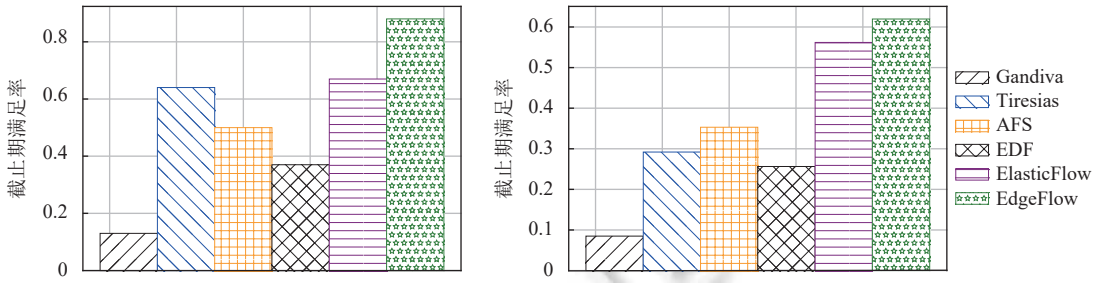
(4) EDF[28]: EDF (earliest deadline first) 是一个面向具有截止期的作业的调度算法, 优先执行截止期最早的作业. EDF 时截止期感知的, 但不是弹性的.

(5) ElasticFlow[13]: ElasticFlow 是一个面向具有截止期的深度学习训练作业的调度器, 可以根据作业的截止期和集群资源紧张程度动态调整每一个作业使用的资源数量. ElasticFlow 是截止期感知的, 且其所调度的作业是弹性的, 但 ElasticFlow 不能利用空闲的边缘服务器资源.

5.3 端到端实验

首先进行端到端实验, 验证 EdgeFlow 的调度方法可以使得更多的作业满足截止期需求. 我们模拟了踪迹 1 和踪迹 2 分别在 8 个和 10 个具有 8 块 NVIDIA A100 GPU 的 GPU 集群上 (共分别具有 64 块 GPU 和 80 块 GPU) 执行的过程. 为了模拟可用空闲边缘服务器数量随时间的变化, 本文模拟器根据边缘服务器利用率的真实数据[5,29], 模拟每天凌晨 2 时至上午 9 时全部边缘服务器空闲、下午 14–17 时有一半的边缘服务器空闲, 在本次实验中边缘服务器的总数量分别设置为 8 个和 10 个. 图 4 展示了 EdgeFlow 和基线方法在两个不同的作业提交历史踪迹下的作业截止期满足率. EdgeFlow 的截止期满足率比基线方法高 1.1–7.3 倍. Gandiva、Tiresias 和 AFS 不是截止期感

知的调度方法, 因此截止期满足率较低; EDF 虽然是截止期感知的, 但其调度方法没有充分利用深度学习训练作业的特点, 在调度深度学习训练作业的场景下截止期满足率的表现欠佳; ElasticFlow 可以根据云计算集群资源紧张程度、每个深度学习作业的特点和每个深度学习作业的截止期, 动态调整作业使用的资源数量, 但该方法无法利用空闲的边缘服务器资源为紧张的 GPU 集群资源分载作业, 因此其截止期满足率仍有提升空间. EdgeFlow 不仅可以灵活、充分利用云计算集群的 GPU 资源, 还可以将合适的作业分载至空间的边缘服务器, 因此在 ElasticFlow 的基础上进一步提升了截止期满足率.



(a) 阿里巴巴集群作业提交踪迹的截止期满足率 (b) Philly 集群作业提交踪迹的截止期满足率

图 4 端到端实验中的截止期满足率

在本节的端到端实验中, 平均每个作业在其生命周期内的迁移次数为 34.9 次, 其中平均云-边迁移次数为 0.6 次. 平均每个作业迁移开销约为 3 min, 与深度学习训练作业执行的总时长相比, 这一开销较小. 而且, 即使存在这样的开销, EdgeFlow 仍可以取得比基线方法更高的截止期满足率.

5.4 消融实验

为了探究 EdgeFlow 实现的不同优化对 EdgeFlow 最终的截止期满足率分别带来多少提升, 我们在第 5.1 节踪迹 1 的实验设定下进行了消融实验, 图 5 展示了消融实验的结果. 其中, ElasticFlow+Edge 指的是用 ElasticFlow 的调度算法进行调度, 并利用可用的边缘服务器资源, 但调度器没有使用第 4.3 节所表述的分载算法选取分载到边缘服务器的作业, 而是将边缘服务器资源和云计算集群中的 GPU 视为同样的算力. 通过图 5 可以看出, 利用空闲的边缘服务器资源可以在一定程度上提高截止期满足率, 但是如果忽略不同作业在不同硬件上执行的性能特性, 截止期满足率并不能得到较大的提升. 只有充分利用可利用的边缘服务器, 并且选择合适的作业分载至这些边缘服务器, 才能对截止期满足率带来更多的提升.

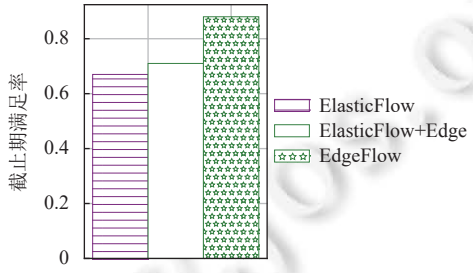


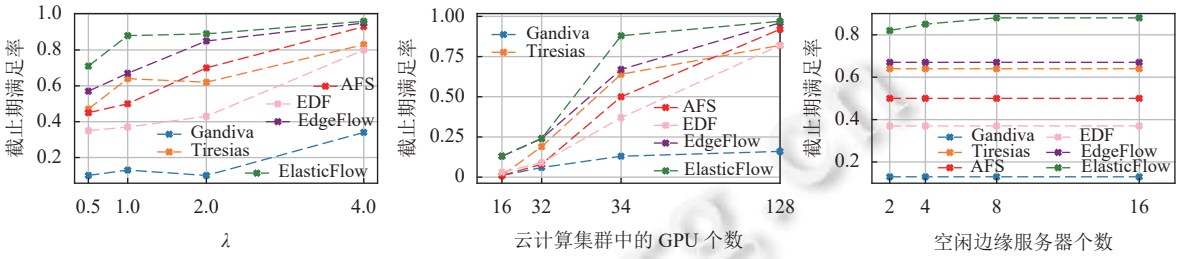
图 5 不同优化对 EdgeFlow 提升截止期满足率的效果提升

5.5 影响因素分析

为了进一步探究 EdgeFlow 在不同因素影响下效果的差异, 我们通过控制变量的方法每次只改变实验设定中的一个变量, 然后模拟作业提交后 EdgeFlow 和其他基线调度器的调度结果.

我们首先探究作业提交时间间隔 (即: 相邻的两个作业提交的提交时间之差) 对 EdgeFlow 的影响. 与现有工作[30]一致, 我们手动生成了具有不同平均作业提交时间间隔的作业提交踪迹. 图 6(a) 展示了在不同作业提交间隔

下调度器的调度结果, 其中横轴表示当前作业提交记录的平均作业提交时间间隔是初始提交记录的 λ 倍. 我们可以看出, EdgeFlow 在不同的作业提交时间间隔下都能保证比基线方法更高的截止期满足率, 且作业提交时间间隔越小, EdgeFlow 对截止期满足率的提升越大. 这是因为作业提交的时间间隔越小, 集群资源和边缘服务器资源越紧张, EdgeFlow 对分载作业的选择越有优势; 而作业提交时间间隔足够大时, 在同样的时间范围内, 云计算集群资源只需要执行更少的训练作业, 云计算集群资源不再紧张, 因此基线调度器也可以使得提交的作业具有较高的截止期满足率.



(a) 作业提交时间间隔的影响 (b) 云计算集群中的 GPU 个数的影响 (c) 空闲边缘服务器个数的影响

图 6 不同因素对 EdgeFlow 的影响

如果将一组同样的作业提交到不同大小的云计算集群上, 云计算集群拥有的计算资源越多, 能在截止期前完成的作业越多. 图 6(b) 展示了踪迹 1 中的作业提交到具有不同 GPU 个数的云计算集群上之后可以满足截止期需求的作业的比例, EdgeFlow 在所有云计算集群大小下都能使得截止期满足率高于基线方法. 在云计算集群非常小时, 大量的作业需要等待已提交的作业完成后才能执行, 即使有可以利用的空闲边缘服务器, 这些边缘服务器所提供的算力也只能执行少量的作业负载, 因此只有少数作业在截止期之前完成. 在云计算集群足够大时, 提交到集群的作业无需等待过多时间即可执行, 因此, 即使不利用空闲的边缘服务器, 也有相当大一部分作业能在截止期之前完成. 在云计算集群中有用的 GPU 数量适中时, EdgeFlow 通过将部分作业分载至边缘服务器, 在缓解云计算集群资源紧张程度的同时让云计算集群资源得到更高效的利用, 从而更充分发挥空闲边缘服务器带来的优势.

空闲的边缘服务器的数量不同, 可被分载至边缘服务器的作业数量不同, 因此会带来不同的截止期满足率. 图 6(c) 展示了踪迹 1 中的作业提交到云计算集群后, 如果可用的空闲边缘服务器数量不同, 分别会有多少作业满足截止期需求. 由于基线方法均不利用空闲的边缘服务器, 图 6(c) 只展示了本文提出的 EdgeFlow 方法的截止期满足率. 在可利用的边缘服务器个数较少时, 只有少量的作业可以被分载至边缘服务器, 因此 EdgeFlow 只能带来少量的优势. 当可利用的空闲边缘服务器数量足够多时, 因为深度学习训练作业在边缘服务器上的训练速度比在 GPU 上的训练速度慢. 为了保证在云计算集群执行可以满足截止期需求的作业不因为被分载至边缘服务器而违背截止期, EdgeFlow 的作业分载算法不会将空闲边缘服务器全部利用起来, 而是只利用算法所需的部分边缘服务器, 因此, 在边缘服务器足够多时, 作业的截止期满足率不会再随着空闲边缘服务器数量的增加而增加. 基线方法无法灵活使用空闲边缘服务器的资源, 截止期满足率不随空闲边缘服务器个数而改变.

6 局限性与未来工作

本文面向的场景是在云计算集群和空闲边缘服务器上调度截止期敏感的深度学习训练作业. 但本文具有一定的局限性, 例如: (1) 本文的优化目标为提升集群中截止期敏感作业的截止期满足率, 而对于有些场景 (例如, 部分作业没有截止期需求、作业有优先级等), 本文方法不适用. 本文算法可以被进一步扩展, 以适配更多的场景. (2) 在实际集群中, 可能会有随机的节点故障. EdgeFlow 当前的设计可能将云计算集群中的全部节点都用于作业的执行, 若节点出现故障, 则可能导致部分作业的截止期无法被满足. (3) 本文为了简化边缘服务器的场景, 同时统一考虑不同边缘服务器的配置和资源使用情况, 只较为保守地为边缘服务器上的每个作业调度一个 GPU. 而在实际场景中, 不同边缘服务器上的空闲 GPU 数量可能不同.

本文下一步工作可以包括: (1) 将本文要解决的问题扩展为一个多目标优化问题, 从而可以使本文的方法适用于更多的场景. 例如, 在作业具有优先级的场景下, 在资源调度顺序满足作业优先级的同时, 提高作业的截止期满足率. (2) 考虑集群中的节点故障和容错问题, 通过计算故障概率并预留部分的云集群资源或边缘节点的算力, 提升系统的容错能力, 从而在节点出现故障的情况下也能尽可能让更多的作业在截止期之前完成. (3) 考虑更多更复杂的情况, 例如, 根据不同边缘服务器上不同的空闲 GPU 数量, 设计边缘服务器上的作业使用多个 GPU 的算法.

7 总 结

边缘服务器在各类生活场景中得到越来越多的重视和使用, 但是由于用户使用边缘服务器的作业负载具有潮汐性的特点, 边缘服务器的计算资源在部分时间段处于闲置状态, 没有得到充分利用. 本文设计云边协同调度系统 EdgeFlow, 将这部分闲置的边缘服务器计算资源用于云计算集群中的分布式深度学习训练作业, 在提高边缘服务器利用率的同时, 使得更多的截止期敏感的深度学习训练作业在其截止期之前完成. 实验结果证明, EdgeFlow 在提升作业的截止期满足率方面优于其他基线方法.

References:

[1] Xu MW, Fu Z, Ma X, Zhang L, Li YN, Qian F, Wang SG, Li K, Yang JY, Liu XZ. From cloud to edge: A first look at public edge platforms. In: Proc. of the 21st ACM Internet Measurement Conf. ACM, 2021. 37–53. [doi: 10.1145/3487552.3487815]

[2] Zhang XM, Zhang AL, Sun JC, Zhu X, Guo YE, Qian F, Mao ZM. EMP: Edge-assisted multi-vehicle perception. In: Proc. of the 27th Annual Int'l Conf. on Mobile Computing and Networking. New Orleans: ACM, 2021. 545–558. [doi: 10.1145/3447993.3483242]

[3] Shi WS, Cao J, Zhang Q, Li YHZ, Xu LY. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637–646. [doi: 10.1109/JIOT.2016.2579198]

[4] Xu MW, Zhang L, Wang SG. Position paper: Renovating edge servers with ARM SoCs. In: Proc. of the 7th IEEE/ACM Symp. on Edge Computing. Seattle: IEEE, 2022. 216–223. [doi: 10.1109/SEC54971.2022.00024]

[5] Xu DL, Xu MW, Lou CH, Zhang L, Huang G, Jin X, Liu XZ. SoCFlow: Efficient and scalable DNN training on SoC-clustered edge servers. In: Proc. of the 29th ACM Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. San Diego: ACM, 2024. 368–385. [doi: 10.1145/3617232.3624847]

[6] Mahajan K, Balasubramanian A, Singhvi A, Venkataraman S, Akella A, Phanishayee A, Chawla S. Themis: Fair and efficient GPU cluster scheduling. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 289–304.

[7] Bérard A, Pietquin O, Besacier L, Servan C. Listen and translate: A proof of concept for end-to-end speech-to-text translation. arXiv:1612.01744, 2016.

[8] Chen CY, Seff A, Kornhauser A, Xiao JX. DeepDriving: Learning affordance for direct perception in autonomous driving. In: Proc. of the 2015 IEEE Int'l Conf. on Computer Vision. Santiago: IEEE, 2015. 2722–2730. [doi: 10.1109/ICCV.2015.312]

[9] Gu JC, Chowdhury M, Shin KG, Zhu YB, Jeon M, Qian JJ, Liu HQ, Guo CX. Tiresias: A GPU cluster manager for distributed deep learning. In: Proc. of the 16th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2019. 485–500.

[10] Hwang C, Kim T, Kim S, Shin J, Park K. Elastic resource sharing for distributed deep learning. In: Proc. of the 18th USENIX Symp. on Networked Systems Design and Implementation. USENIX Association, 2021. 721–739.

[11] Xiao WC, Bhardwaj R, Ramjee R, Sivathanu M, Kwatra N, Han ZH, Patel P, Peng X, Zhao HY, Zhang QL, Yang F, Zhou LD. Gandiva: Introspective cluster scheduling for deep learning. In: Proc. of the 13th USENIX Symp. on Operating Systems Design and Implementation. Carlsbad: USENIX Association, 2018. 595–610.

[12] Gao W, Ye ZS, Sun P, Wen YG, Zhang TW. Chronus: A novel deadline-aware scheduler for deep learning training jobs. In: Proc. of the 2021 ACM Symp. on Cloud Computing. Seattle: ACM, 2021. 609–623. [doi: 10.1145/3472883.3486978]

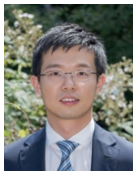
[13] Gu DD, Zhao YH, Zhong YM, Xiong YF, Han ZH, Cheng P, Yang F, Huang G, Jin X, Liu XZ. ElasticFlow: An elastic serverless training platform for distributed deep learning. In: Proc. of the 28th ACM Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. Vancouver: ACM, 2023. 266–280. [doi: 10.1145/3575693.3575721]

[14] Weng QZ, Xiao WC, Yu YH, Wang W, Wang C, He J, Li Y, Zhang LP, Lin W, Ding Y. MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters. In: Proc. of the 19th USENIX Symp. on Networked Systems Design and

Implementation. Renton: USENIX Association, 2022. 945–960.
[15] TorchElastic. 2022. https://pytorch.org/elastic/0.2.0/index.html
[16] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: Proc. of the 2009 IEEE Conf. on Computer Vision and Pattern Recognition. Miami: IEEE, 2009. 248–255. [doi: 10.1109/CVPR.2009.5206848]
[17] He KM, Zhang XY, Ren SQ, Sun J. Deep residual learning for image recognition. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 770–778. [doi: 10.1109/CVPR.2016.90]
[18] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2015.
[19] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition. Las Vegas: IEEE, 2016. 2818–2826. [doi: 10.1109/CVPR.2016.308]
[20] Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning word vectors for sentiment analysis. In: Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland: Association for Computational Linguistics, 2011. 142–150.
[21] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. 2019. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
[22] Panayotov V, Chen GG, Povey D, Khudanpur S. LibriSpeech: An ASR corpus based on public domain audio books. In: Proc. of the 2015 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing. South Brisbane: IEEE, 2015. 5206–5210. [doi: 10.1109/ICASSP.2015.7178964]
[23] Amodei D, Ananthanarayanan S, Anubhai R, *et al*. Deep Speech 2: End-to-end speech recognition in English and Mandarin. In: Proc. of the 33rd Int'l Conf. on Machine Learning. New York: JMLR.org, 2016. 173–182.
[24] Zhao HY, Han ZH, Yang Z, Zhang QL, Yang F, Zhou LD, Yang M, Lau FCM, Wang YQ, Xiong YF, Wang B. HiveD: Sharing a GPU cluster for deep learning with guarantees. In: Proc. of the 14th USENIX Symp. on Operating Systems Design and Implementation. USENIX Association, 2020. 515–532.
[25] Microsoft Philly Trace. 2019. https://github.com/msr-fiddle/philly-traces
[26] Amazon EC2 P4 Instance. 2024. https://aws.amazon.com/cn/ec2/instance-types/p4/
[27] Koumoutzelis S, Giannoulakis I, Georgoulakis T, Avdikos G, Kafetzakis E. Security issues of GPUs and FPGAs for AI-powered near & far edge services. In: Proc. of the 22nd European Conf. on Cyber Warfare and Security, 2023. 703–706. [doi: 10.34190/eccws.22.1.1160]
[28] George L, Rivierre N, Spuri M. Preemptive and non-preemptive real-time uniprocessor scheduling. 1996. https://inria.hal.science/inria-00073732/PDF/RR-2966.pdf
[29] Malandrino F, Kirkpatrick S, Chiasserini CF. How close to the edge? Delay/utilization trends in MEC. In: Proc. of the 2016 ACM Workshop on Cloud-assisted Networking. Irvine: ACM, 2016. 37–42. [doi: 10.1145/3010079.3010080]
[30] Narayanan D, Santhanam K, Kazhamiaka F, Phanishayee A, Zaharia M. Heterogeneity-aware cluster scheduling policies for deep learning workloads. In: Proc. of the 14th USENIX Symp. on Operating Systems Design and Implementation. USENIX Association, 2020. 481–498.

谷典典(1998－), 女, 博士, 主要研究领域为机器学习系统, 集群调度, 分布式机器学习.

刘譞哲(1980－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为服务计算, 系统软件.



金鑫(1989－), 男, 博士, 副教授, CCF 高级会员, 主要研究领域为系统软件, 计算机网络, 云计算.