未来网络发展大会
FUTURE NETWORK
DEVELOPMENT CONFERENCE
NANJING · CHINA / 中国 · 南京

未来网络技术发展系列白皮书（2025）

AI大模型跨域训练池化调度技术体系白皮书

第九届未来网络发展大会组委会

2025年8月

版权声明

编写说明

主要编写单位：

紫金山实验室、江苏省未来网络研究院

主要编写人员：

周俊、孙远、刘准、张晨、高新平、杨彩云、孙婵娟、王春生、肖玉明、梁木

特别鸣谢：

新华三、天数智芯、浪潮信息、中兴通讯、中国电信

前　言

AI 大模型的跨域训练是全球范围关注的前沿技术方向，它是指将多个不同的智算中心组合在一起训练同一个 AI 大模型。为什么需要跨域训练？业界通常的认知在于，当大模型未来发展到万亿、十万亿参数规模时，根据 Scaling Law 需要用到万卡甚至十万卡才能完成其预训练过程，这样的体量规模如果集中到一个集群内部，在技术、能源、配套等方面都存在着严峻的挑战，因此需要通过网络连接多个集群并加以组合，以共同训练同一个万亿/十万亿的大模型。

实际上自 OpenAI 发布 GPT-4 后，业界就一直在围绕下一代 GPT 的需求进行跨域训练的探索。这样的认知与实践自然无可厚非，它更多地关注于通用大模型的发展问题，是一种“少数人的游戏”。DeepSeek 发布 V3/R1 后，通用大模型不可逆地走上了开源路线，原有牌桌上“少数人”中的大部分又被迫离场，目前已变成了“几个玩家的游戏”。

一个令人焦虑的问题是，虽然打牌的人越来越少，但牌桌却无法自动缩小反而仍在不断扩大，这于我国而言是十分明显的。根据国家数据局最新数据，我国算力总规模已排在全球第二位，但由于诸多方面的原因，我国的高端智能算力领域却同时面临着“少、杂、散”的客观困境。破局点在哪？让我们重回 2023 年底《关于深入实施“东数西算”工程加快构建全国一体化算力网的实施意见》（简称意见），意见在基本原则中明确指出“充分发掘重点行业算力需求，盘活存量

算力资源”,“探索异属异构异地的算力资源并网调度技术方案和商业模式”，其中即蕴含了破局之道。

DeepSeek 开源后，虽然通用大模型的玩家廖然无几，但却极大地带动了行业的算力需求，企业不仅可使用“DeepSeek+知识库”进行推理，还可以基于“DeepSeek+数据集”通过后训练培养出自身专属专用的“企业大模型”。与通用大模型“广而杂”不同，“企业大模型”需要的是“专而精”，百亿级参数通常足够日常生产使用，一次后训练的算力需求大多在几十卡的规模，卡的型号并不追求高端顶尖，出于成本考虑这些企业通常也不会为后训练自建集群，通过租用算力会更经济实惠。与通用大模型预训练“开一单、吃三年”的“算力房地产”模式不同，企业大模型后训练更适合薄利多销、细水长流的“算力网调度”模式，在全国一体化算力网的服务能力加持下，千行百业按需消纳“异属异构异地”的存量算力资源，把“少数人的游戏”变回“一群人的生态”。

《AI 大模型跨域训练池化调度-技术体系白皮书》（简称白皮书）的编制，是基于未来网络团队多年来在 AI 大模型跨域训练与算力网调度方面结合实践的创新成果。与业界面向于通用大模型在“同属、同构/异构、同城/异地”资源上的拉远部署技术路线有所不同，未来网络专注于企业大模型在“异属、异构、异地”资源上的池化调度技术路线，通过“广域确定性网络+智算资源并网+算网协同调度”三位一体的技术架构，可真正实现“异属合训、异构混训、异地同训”的池化调度能力。

白皮书围绕技术体系视角，对于 AI 大模型跨域训练池化调度的参考架构、关键技术、试验评估等进行了详细的介绍。希望能够通过本白皮书的内容，为业界树立基于“异属异构异地”资源的 AI 大模型跨域训练池化调度范式，为实现全国一台超级计算机的宏伟目标走出未来网络创新路径。

目　　录

一、 现状与挑战

1.1 AI 大模型

“训练—推理”这一范式脱胎于早期的深度学习模型，CNN、DNN、RNN 等 AI 模型等虽已具备模型训练、参数优化的框架，但其规模相对有限，训练通常使用单机单卡或单机多卡即可完成。与之相比，AI 大模型的核心特征即在于其庞大的参数量（通常达到百亿、千亿乃至万亿级别）和基于超大规模数据集（TB 级别）的训练，这一过程所需的 GPU 核心和显存资源远超单机承载能力，对分布式并行计算架构提出了前所未有的极高要求。

早期模型的分布式并行计算架构通常采用中心化的数据并行架构，以 1 个参数服务器（PS，Parameter Server）为总协调控制 N 个工作节点（WN，Worker Node）并行计算，流行于 TensorFlow 框架的开发生态。随着 GPT 类大模型的发展，去中心化的 3D 混合并行架构（DP 数据并行、TP 张量并行、PP 流水线并行）得到广泛应用，PyTorch 也逐步取代 TensorFlow 成为业界事实标准。GPT-4 的问世，将专家并行叠加于 3D 并行之上形成混合专家架构（MoE，Mixture of Experts）。不久前 GPT-5 发布，据有关预计其参数量已达到十万亿量级。

如此大规模的模型，来源于全球对于通用人工智能（AGI，Artificial General Intelligence）的狂热追求，以及扩展法则（Scaling Law）

的持续作用。它们基于互联网上爬到的数据进行训练，要花费成千上万张甚至数十万的 GPU 资源才能训练出来，虽然可以陪人闲聊、回答问题甚至求解方程，但却无法知道的企业流水线的工艺制造方式、学校对学生的个性培养计划、医院为老人的病症诊疗方案。这些大模型被称为“通用大模型”，它知道的很多很杂、但不深不准。如果要让大模型真正服务于千行百业，需要的是把“通用大模型”与行业数据充分结合，再通过算力加工成“行业大模型”。

目前，“行业大模型”的发展正处于初期阶段，DeepSeek-V3/R1 在年初的开源，使得动辄千万的商用大模型成本直降为 0，企业真正享受到了“大模型平权”：不仅可使用“DeepSeek+知识库”进行推理，还可以基于“DeepSeek+数据集”通过后训练培养出自身专属专用的“企业大模型”。与通用大模型“广而杂”不同，“企业大模型”需要的是“专而精”，百级参数通常足够日常生产使用，一次后训练的算力需求大多在几十卡或百卡的规模，卡的型号也并不追求高端顶尖。出于使用频次和成本考虑，企业自身通常也不会为后训练自建本地集群，通过租用算力会更经济实惠。

然而一个客观的情况是，企业并不情愿到公有云上租用算力，简而言之就是“数据传不出、网络运不动、算力信不过”，因此年初 DeepSeek 爆火后业界发展一体机的形态更符合现实的需求，但一体机通常只能推理无法训练，企业只能靠知识库“查字典”，而无法学习数据集“举一反三”。行业大模型的发展之路，仍存在巨大挑战。

1.2 跨域训练

AI 大模型的跨域训练，是指将一个大模型的训练任务切分到多个智算中心进行协同训练。在大模型出现前，一类面向 PS 架构较为常见的实现，是对模型采用“分级部署+数据压缩+异步训练”的思路，将数据集拆分到多个智算中心进行跨域数据并行。大模型出现后，面向 3D 并行架构的思路是将模型参数拆分到不同智算中心，根据不同的拆分方法，可将 DP 或 PP 流量基于广域网进行传输，而 TP 和 EP 要求超低时延、巨大带宽通常只能在智算中心内部、甚至智算服务器内部通信。

跨域训练是全球范围关注的前沿技术。于我国而言，由于智算的产业生态存在着诸多特殊之处，跨域训练也面临着额外的挑战。简言之，我国在高端智算领域正面临着“少、杂、散”的客观困境：（1）受限于 AI 芯片封锁政策，英伟达等先进型号的 AI 芯片极为稀缺并进一步被各路运营渠道所瓜分；（2）国产 AI 芯片 GPGPU、NPU、ASIC 等不同技术路线并行发展，同时受英伟达高速发展的牵引不断衍生出各种型号；（3）诸多地方政府将智能算力纳入公共基础设施建设，AI 芯片通过各地基建项目被分流至不同城市或区域。针对于此，《关于深入实施“东数西算”工程加快构建全国一体化算力网的实施意见》（简称意见）中，提出“探索异属、异构、异地算力资源并网调度的技术方案与商业模式”，以此寻求破局之道。

“异属”即各智算中心运营主体是多元化的，如不同市场性质的

企业、不同行政管辖的园区、不同的高校科研院所等；**“异构”**即智算中心间技术架构上是差异化的，如所用服务器中智算芯片的厂家、架构、型号，内部网络的拓扑与协议，集群软件的管理与控制等；**“异地”**即智算中心在地理位置上是分散化的，如位于同一城市的不同园区、同一省份的不同城市、不同省份乃至跨东西部区域等。“三异”可组合出多种情况，技术上最简单的情况是单个智算中心的“同属、同构、同地”，最复杂的情况是多个智算中心的“异属、异构、异地”，而我国的 AI 大模型跨域训练就面临着“异属、异构、异地”的巨大技术挑战：

“异属”挑战在于各主体彼此独立规划、建设并运营自身的智算中心，当这些智算中心并入算力网并运行同一个训练任务的不同部分时，由于各资源自身的内部网络规划、管理控制平台、对外运营服务等方面存在着巨大的差异化甚至冲突性；

“异构”挑战在于不同厂家、不同架构、不同型号智算芯片间的适配问题，当同一个训练任务的不同部分运行在多种智算芯片之上，由于各芯片自身的算力/显存大小、互联拓扑/性能、算子库/通信库等方面存在诸多的差异性；

“异地”挑战在于当多个位于不同城市或区域的智算中心运行同一个训练任务的不同部分时，智算中心之间的网络传输带宽、时延/抖动、丢包/乱序等问题会对并行流量产生不同程度的影响；

上述“异属”、“异构”、“异地”中任意问题，都可能会导致训练任务执行效率的大幅下降甚至无法运行，而“异属”、“异构”、“异地”

的组合，更加剧了问题的严峻性。

1.3 池化调度

在 DeepSeek 开源之前，通用大模型预训练是一种可称为“算力房地产”的生态模式，算力的供需双方线下签订合同并交付资源，通过手动部署调优运维的方式开展训练过程，而线上更多是一种过单的操作形式。DeepSeek 开源后，通用大模型的玩家骤然减少，很多“算力房地产”的“模型入住率”大大降低。当大量算力资源被释放之后，如何能够通过“算力网调度”对算力资源进行在网的动态消纳，就成为了亟待解决的迫切问题。

业界目前讨论更多的是“算力调度”，何为“算力网调度”？这里，需要先对“算力调度”正本清源。随着东数西算工程与全国一体化算力网的浩荡展开，各类所谓的“调度平台”竞相上岗，深藏在规模建设后的是技术路线的鱼龙混杂。目前来看，大体有以下几类：

（1）云计算门户（传统），其业务本质是“用户自选”（供应商/地域/卡型），“歧视定价”（目录价高/线下折扣/周期返点），其商业本质是“算力自营”（自建自销），“纯供方市场”（供方对定价达成联盟协议）；

（2）多云管理工具（过渡），其业务本质是“用户自选”（供应商/地域/卡型）、“代开代维”，其商业本质是“算力管理”，严格说其应属一种工具而不是一种运营模式，为需方提供了管理便利，但不改变云计算的供方市场格局；

（3）**算力交易电商（现状）**，其业务本质是“信息集中公开”（规格/定价）、“供需交易撮合”（过单/抢单），其商业本质是“算力中介”，随引入渠道服务有利于转为需方市场，但不具备调度能力；

（4）算力调度（演进），其业务本质是“任务式服务”（目录价低/按需启停/精准计量/效用付费）、“租机器调任务”，其商业本质是“算力经销”，即先批发再加工转零售，本质上仍属于供方市场且弱化了渠道属性；

（5）算力网调度（目标），其业务本质是“任务式服务”（最优匹配/按需启停/精准计量/效用付费）、“调度推荐”（交互式调度/算网协同/全流向调度），其商业本质是“算力分销”，通过调度连接强化渠道服务，充分向需方市场引导。

东数西算的终极形态，需要算力网调度来保障支撑。目前，业界对于算力网调度技术的探索刚刚起步，“三异”资源的封装抽象尚未有成功案例。虽然像“用水、用电”一样“用算”已成为大家经常谈起的目标，但实际上我们在智算领域面对的仍然是“多口小水井”而不是“一汪大水池”。如何能够让一个 AI 大模型通过调度系统自动调动、分发到多个“异属异构异地”智算中心去训练，而无需用户关心归属、架构、位置，在全球范围尚无先例。

二、 技术路线分析

AI 大模型跨域训练，业界目前已有诸多实践。国内三大运营商从

异地角度切入，纷纷基于其新型广域网能力开展了多样化探索，从同城数十公里、跨城数百公里、跨区域千公里级，逐步强化异地尺度。国内有关模型与芯片公司，从异构角度切入，开展了国产卡与英伟达卡间的异构管理、混合训练。近期，上海人工智能实验室联合中国电信、中国联通发布了跨 1500 公里的异构混训成果，标志着业界对于算力网池化的认知逐步升级。但较为遗憾的是，业界虽有较多发声，但对于其中的细节却甚少披露。

不过，从有限的信息中依然能够管中窥豹。无论业界前期的何种尝试，从全国一体化算力网的角度而言，均尚停留在“异地”和“异构”层面，“异属”均未涉及。同时，业界绝大部分实践都是基于手动配置调优，鲜有端到端全流程自动化调度。本节首先分析业界应用场景与技术路线，同时给出未来网络的应用场景与技术路线，以便读者在进入后续技术章节之前，能够更加清晰地把握个中逻辑。

2.1 专用算力拉远

对于 AI 跨域训练，业界的主流认识来自于这样一个预测：当大模型未来发展到万亿、十万亿参数规模时，根据 Scaling Law 需要用到万卡甚至十万卡才能完成其预训练过程，这样的体量规模如果集中到一个集群内部，在技术、能源、配套等方面都存在着严峻的挑战，因此需要通过网络将多个集群进行连接，以协同训练同一个万亿/十万亿的通用大模型。

基于这种认识，业界在 AI 跨域训练中所采用的技术路线，可以

理解为主要是面向通用大模型场景。不过，客观而言这是一种“少数人的游戏”，尤其在 DeepSeek 开源之后，目前玩家已所剩无几。不过，考虑到 AGI 的战略意义，这种探索实践仍具有重大价值。

这种业务场景的特征在于：（1）基础设施为通用大模型所专用，对于万卡/十万的规模体量而言，基础设施上运行这个一个大模型即完全足以“开一单吃三年”，合同签订后一手交钱一手交货，用户自身对于基础设施的理解比算力服务商可能还要高出 1-2 个段位，平台的标准化交付反而会拖累模型性能，而且动辄千万甚至过亿的成交额，也不适合基于平台线上成交；（2）用户通常会精心选择所用的智算中心，即使需要跨域也会尽可能地选择“同城、同构”的机房，不会为了“异地、异构”而舍近求远、舍本逐末，异属就更加不会考虑；（3）智算中心的数量会控制在 3 个以内，以控制系统复杂度。

因此，对于用户和算力服务商而言：算力专用于特定的大模型，所有算力资源最好能在同一个机房，如果确实难以实现会尽量把服务器就近搬到同属、同构、同城的机房。

技术路线方面，上述业务场景意味着 2 到 3 个特定智算中心的点对点互联，集群管理的服务器直通，以及模型部署的透明化运行：**（1）网络方面**，通过光纤直驱或者 OTN 电路交换提供点对点的硬管道带宽；**（2）调度方面**，无论智算中心分布均以一套 K8S 类管理系统直接管理服务器资源；**（3）模型方面**，单个集群内部的 Megatron 等框架最好完全透明地移植到新的环境之上。

因此，上述业务场景与技术路线，我们将其称为“专用算力拉远”，

本质上是对于单集群本地训练的“环境复刻”，这是一种纯商业驱动的市场行为，算力网的生态模式显然并不在其考虑范围之内。

2.2 全局池化调度

与通用大模型预训练“开一单、吃三年”的“算力房地产”模式不同，企业大模型后训练更适合薄利多销、细水长流的“算力网调度”模式，在全国一体化算力网的服务能力加持下，千行百业按需消纳“异属异构异地”的存量算力资源，把“少数人的游戏”变回“一群人的生态”。

这种业务场景的特征在于：（1）基础设施被不同用户的企业大模型所复用，企业大模型的一次后训练，可能就是几十张卡训练 3 天，用户自身对于基础设施的理解基本为 0，最好能通过平台进行“傻瓜式”的操作；（2）用户并不关心某次训练所用的智算中心，无论是哪一家供应商、使用何种处理器架构、跨越多远的距离，能够快速、便宜地把模型训练出来，是用户唯一关心的问题；（3）不直接排斥在一次训练任务被调度到多个智算中心，只要对速度影响不大、不会增加太多额外成本，就都可以接受。

因此，对于用户和算力服务商而言：用户希望能够屏蔽掉与底层资源有关的任何细节，只要模型精度、训练时间、算力成本有所保障；算力服务商愿意尝试与其他服务商训练同一个用户训练任务，只要模型能够无障碍地运行、算力费用能够清晰划分。

技术路线方面，上述业务场景意味着“异属异构异地”的“全局

池化调度”：**（1）网络方面**，需要通过全互联的路由交换网络实现一线接入全局可达，同时需要保障延迟、带宽与丢包、抖动；**（2）调度方面**，需要实现分层跨域的调度结构，以解决异属的跨运营主体调度以及异地的算网协同调度，同时需兼顾异构算力的自动适配；**（3）模型方面**，尽可能地降低跨域传输的数据体量，必要时需实现大模型框架与广域网络的联动优化。

因此，上述业务场景与技术路线，我们将其称为“全局池化调度”，本质上是将全局“三异”资源进行统一抽象，对于用户提供无差别使用，将“多口小水井”变为“一汪大水池”，真正实现“用水用电一样用算”的目标算力网生态。

三、 AI 大模型跨域训练池化调度

3.1 总体架构

大模型跨域训练池化调度架构如图 1 所示，整体呈现出分层解耦的设计理念，可划分为业务层、管控层、资源层三大核心层级。其中，业务层作为需求入口，负责接收各类大模型训练任务请求，并将其转化为标准化的任务描述；管控层作为架构中枢，通过协同调度机制和资源编排策略，实现跨域资源的统一管理与池化调度、大模型作业的拆分与部署；资源层则作为算力底座，整合分散在不同地域的数据中心、云平台等异构计算资源，为训练任务提供可预期的算力支持。这

三层架构相互协作，形成高效的闭环调度体系，有效提升大模型跨域训练的资源利用率和训练效率。

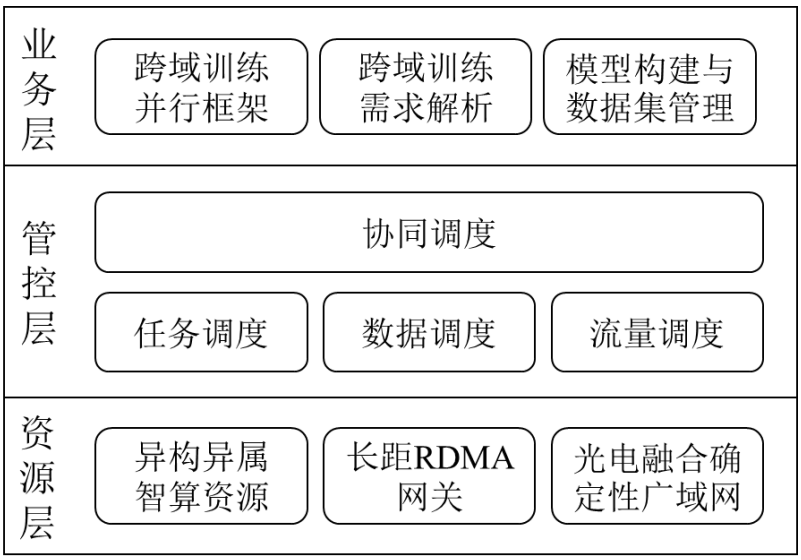

图 1　总体架构

业务层聚焦跨域训练任务资源需求与有限供给的适配难题，核心技术包括大模型跨域训练框架、需求解析、模型与数据集管理等，可将训练任务切分为适配异地异构资源的子任务，动态调整并行策略以降低跨域通信依赖，提供 “一次提交、全域执行”接口。

管控层针对多主体资源协同调度与全局优化难题，涵盖协同调度、算力调度、存储调度、网络调度等技术，旨在打破异属异地资源权属与管理边界，通过统一资源视图实现跨域算网存资源协同匹配。

资源层围绕异构硬件兼容互通与长距通信高效可靠难题，由异构智算资源、长距 RDMA、光电融合确定性广域网等构成，构建“算力—存储—网络”一体化跨域资源底座，屏蔽硬件异构性，突破广域网瓶颈，提供高可靠、低抖动的底层支撑。

大模型跨域训练池化调度技术体系是一套面向大模型跨域训练

场景的系统性解决方案，部署拓扑结构示意图如图 2 所示。

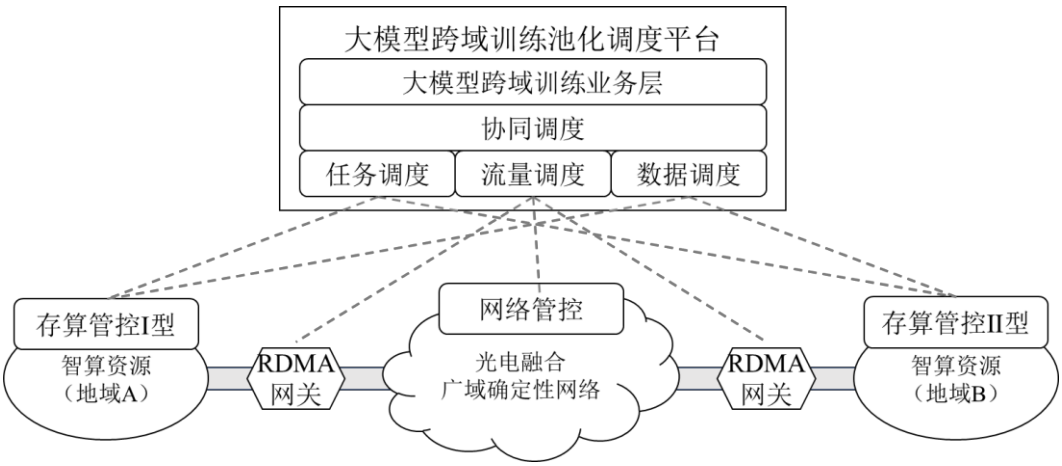

图 2　系统拓扑结构示意图

多地的智算资源通过算力并网接入广域网，形成智算资源算力网拓扑结构。智算资源物理并网时，可在智算资源与广域网间部署 RDMA 网关，以提供长距 RDMA 通信能力。与此同时，智算资源逻辑并网时，智算中心的存算管控将分别于大模型跨域训练平台的任务调度、存储调度的接口对接，提供至上而下的任务、数据调度能力。广域网的网络管控和 RDMA 网关将均与平台的网络调度对接，以提供跨域智算资源的高质量网络服务能力。

3.2　计算通信重叠的跨域训练框架

在大模型跨域训练场景中，通信效率是制约训练性能的关键瓶颈，通过计算与通信重叠流水线、非阻塞 GPU 通信及流水并行通信量优化等技术的协同应用，可显著降低跨域通信对带宽的依赖，提升整体训练效能。这三项技术从时序优化、资源隔离、数据精简三个维度形成互补，在跨域场景中协同降低通信对训练效率的制约，为大模型跨

数据中心联合训练提供了关键技术支撑。

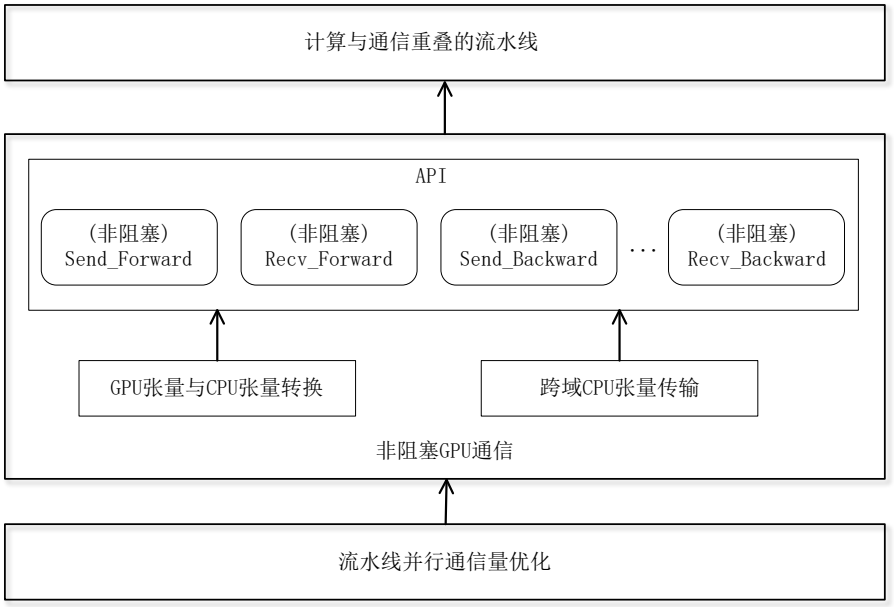

图 3　计算通信重叠的跨域训练框架

（1） 计算与通信重叠流水线模块

计算与通信重叠流水线技术通过精细化的任务拆解与时序编排，在流水线稳定运行阶段实现所有 GPU 通信操作与计算过程的完全掩盖。其核心在于将大模型训练任务按层拆解为连续的子任务单元，当某一层计算在当前 GPU 完成后，立即启动该层参数向后续节点的传输，同时下一层计算在本地 GPU 同步启动，使通信操作嵌入计算间隙，避免因等待数据传输而产生的空闲时间。在跨域场景中，这种机制能将原本串行的“计算—通信”过程转化为并行流，理论上可将跨域通信对整体训练时长的影响压缩至趋近于零，从而大幅降低对广域链路带宽的需求。

（2） 非阻塞 GPU 通信模块

非阻塞 GPU 通信技术通过硬件资源隔离与异步执行机制，实现

通信与计算的完全并行，同时消除资源竞争。该技术依托 GPU 架构中的独立通信引擎（如 NVIDIA 的 GPU Direct RDMA），使数据传输操作可在计算核心执行训练任务时独立运行，且二者分别占用不同的显存分区与 PCIe 通道，避免传统阻塞模式下的资源争抢。在异构跨域环境中，这一特性可确保 AMD MI300 与昇腾 910 等不同架构 GPU 在执行计算密集型任务时，同步完成与远端节点的梯度交换，既提升了单 GPU 的资源利用率，又减少了跨域通信的累积延迟。

（3） 流水线并行通信量优化模块

流水并行通信量优化技术通过重构流水线内的数据交互模式，将跨域通信量压缩至最小粒度。传统流水并行中，每一层计算完成后需向所有后续节点广播完整参数张量，而该技术通过建立层间依赖图谱，仅在相邻阶段间传输必要的中间结果，使单次跨域通信的张量体积缩减为原有的 1/N（N 为流水线阶段数）。例如，在千亿参数模型的跨域训练中，通过将 Transformer 层拆分为 16 个流水阶段，每次跨域传输仅需发送单一层的注意力权重或 FFN 输出，配合张量压缩算法，可使跨域通信带宽需求降低，显著缓解广域网的传输压力。

3.3 跨广域的算网存协同调度

大模型训练作业跨广域网协同调度架构示意图如图 4 所示。管控层由协同调度、任务调度、数据调度、流量调度构成。资源层由算力管控、存储管控、网络管控构成。

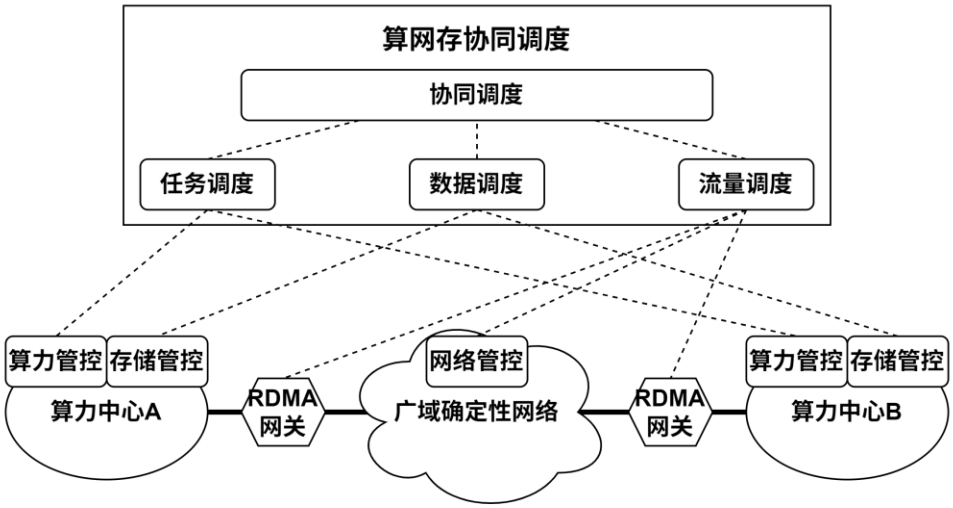

图 4　跨广域的算网存协同调度

（1） 协同调度

承接大模型训练作业跨广域调度请求，是跨广域调度的业务入口。将大模型需求进行解析为算力、网络、存储需求，并以大模型的需求驱动任务调度、数据调度、流量调度的协同工作。

（2） 任务调度

具备处理大模型训练作业的 GPU 算力需求的能力，为大模型训练作业分配合适的算力资源。可将大模型训练作业的 GPU 算力需求拆分到多个算力中心，实现大模型训练作业的跨多算力中心部署。

（3） 数据调度

具备处理大模型训练作业数据访存需求的能力，为大模型训练作业的训练数据集、检查点、模型参数文件等数据分配合适的存储资源。可与任务调度协作，将大模型训练作业的训练数据集、检查点与模型参数文件同步到合适的多个算力中心。

（4） 流量调度

具备处理大模型训练作业通信流量需求与传输流量需求的能力，为大模型训练作业的任务间通信与数据传输分配合适的网络资源。可与任务调度、数据调度协作，为大模型训练作业的任务间通信与数据传输开通确定性网络路径，保障通信与传输的服务质量。

（5） 算力管控

算力管控管理算力中心内的算力资源，可承接来自于任务调度的算力需求。算力管控为任务调度提供其所在算力中心的算力资源状态，作为任务调度的依据。

（6） 存储管控

存储管控管理算力中心内的存储资源，可承接来自于数据调度的存储需求。存储管控为数据调度提供其所在算力中心的存储资源状态，作为数据调度的依据。

（7） 网络管控

网络管控管理其所在广域确定性网络的网络资源，可承接来自于流量调度的网络需求。网络管控为流量调度提供其所在广域网的网络资源状态，作为流量调度的依据。

3.4 异属异构智算资源池化并网

在大模型跨域训练场景中，异属（分属不同机构、企业或主体）、异构（涵盖不同架构的 GPU、CPU、AI 加速芯片等）的智算资源呈现高度分散态势，难以形成高效协同的算力集群。为突破资源壁垒，实现全域算力的统筹利用，亟需构建一套统一、灵活且高效的智算资

源池化并网体系。通过算力资源池化并网实现异属异构异地的算力资源通过网络连接实现算力资源的可达、可用，并通过 API 接口实现算力资源的管理、调度与计量，为大模型跨域训练等场景提供全域协同的坚实算力支撑。

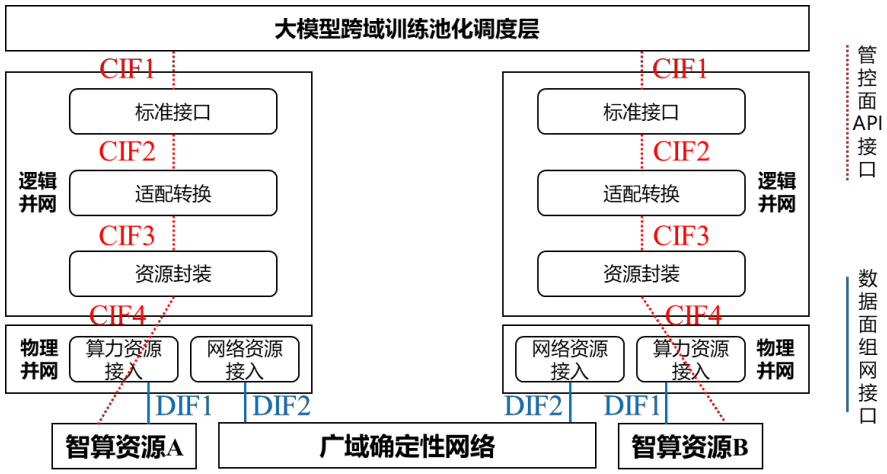

图 5 智能算力并网功能架构图

算力并网功能架构图如图 5 所示。逻辑并网由资源封装、适配转换、标准接口三个功能分层构成，通过逻辑并网端点发生作用。其核心在于依托资源能力封装、功能适配转换与标准接口建模等技术手段，将算力资源抽象为可供平台进行标准化调用的服务能力，进而与平台间实现平台账号/资源监测的标准化管理，以及业务的标准化调度与计量。物理并网分为算力资源接入与网络资源接入，通过物理并网锚点发生作用。其中，算力资源接入实现算力资源与物理并网锚点间的组网连接，对于物理并网锚点而言算力资源接入属于用户侧接口（UNI）；网络资源接入实现物理并网锚点与算力网中网络资源间的组网连接，对于物理并网锚点而言网络资源接入属于网络侧接口（NNI）；基于算力资源接入与网络资源接入，物理并网锚点对用户业

务、平台管控等流量进行路由中继与隔离，进而实现算力资源的可达、可用。

如图 6 所示，逻辑并网的内涵是实现算力资源在账号、监测、调度、计量等功能方面接入平台的整体过程。**横向分层：**算力资源通过资源封装以 API 接口形式提供能力，算力资源经过资源封装后通过适配转换实现标准接口对齐、算力资源经过适配转换后以标准接口的接口规范对接平台。**纵向服务：**账号服务实现用户的认证授权等能力、监测服务实现智算资源信息（如总量/余量等）的上报调取等能力、调度服务实现业务的开通部署等能力（如容器/作业等）、计量服务实现业务的计量等能力。面向 AI 大模型跨域训练场景，需提供作业队列模式的调度服务，以支撑大模型作业跨多队列协作与同步训练。

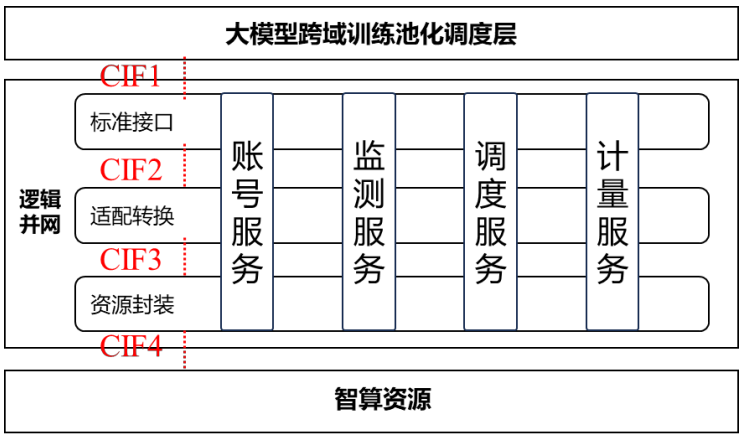

图 6　逻辑并网核心架构

物理并网的内涵是通过多样化的组网连接传输技术连接算力网中的算力资源与网络资源，打通算力网中的用户业务、平台管控等流量传输的端到端连通性，以实现算力资源的可达、可用。物理并网由算力资源接入和网络资源接入构成，智算资源物理并网流量承载见图

7。面向 AI 大模型跨域训练场景，物理并网锚点需具备支撑大模型训练业务流量的 RDMA 传输加速与网络虚拟化等能力，解决异地异属核心技术问题。

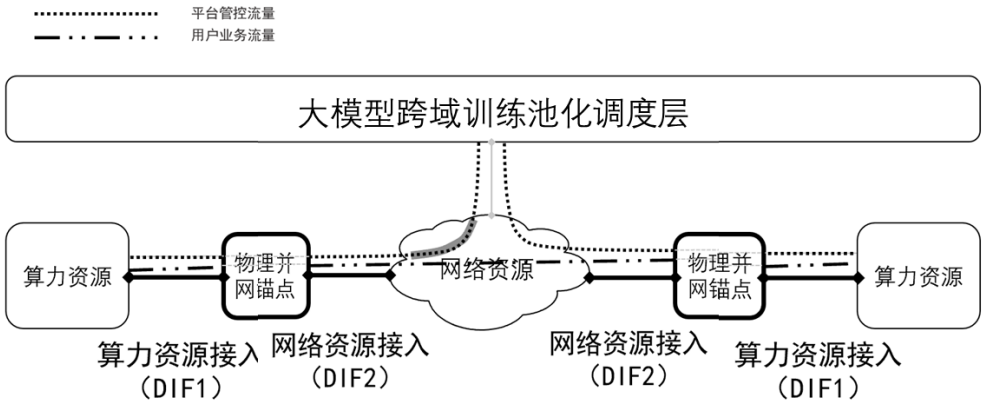

图 7　物理并网流量承载示意图

3.5　光电融合广域确定性网络

广域网面临容量受限、QoS 难承等挑战，难以提供“按需定制”的服务能力，其根本原因在于光传送与数通领域长期独立发展，未能形成有效合力。光电融合确定性广域网重点解决融合组网与灵活调度问题，通过底层全光互联实现大容量长距离的广域传输，融合光电域的多资源维度与全颗粒调度能力，实现资源池化并提供弹性化承载通道，同时构建面向分组的端到端确定性传输能力。

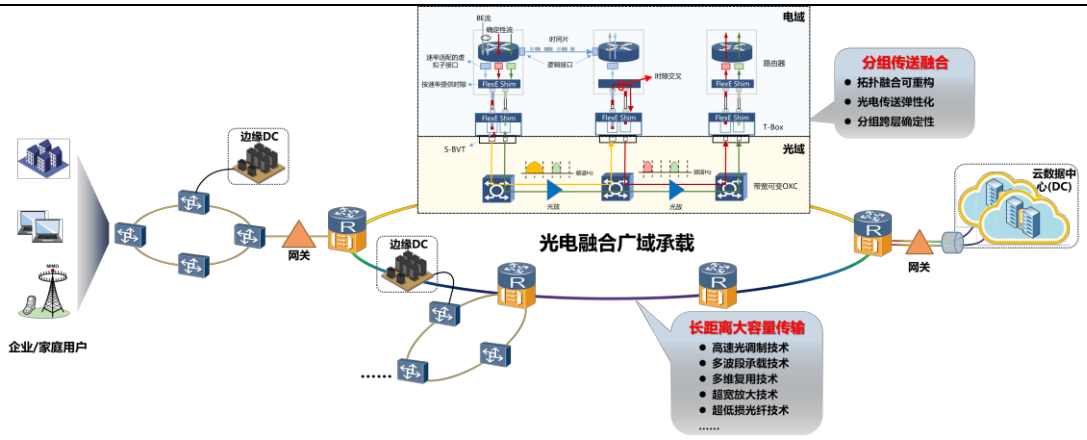

图 8　光电融合广域确定性网络架构

光电融合确定性广域网总体架构如图 8 所示，包括网络控制平面与基础设施平面。其中，控制平面由决策中枢与域控制器组成，承担业务跨域跨层规划与资源调度控制等任务；基础设施平面由光电转发设备与网关组成，形成“分组+TDM+光”的多层融合转发模式，并通过网关衔接不同自治域提供跨域 QoS 定制能力，实现用户与用户、用户与云/边数据中心间的高质量传输控制。光电融合确定性广域网重点解决光电融合组网与灵活调度问题，通过底层全光互联实现大容量长距离的广域传输，融合光电域的多资源维度与多颗粒调度能力，实现资源池化并提供弹性承载通道，构建面向分组的端到端确定性传输能力，最终围绕用户要求提供多维 QoS 量化可承诺的分组传送能力，实现长距离大容量、确定性、弹性化的高效传输控制。

光电融合确定性广域网通过在域内构建基于“分组+TDM+光”的多层组网结构，突破传统物理接口的容量边界，实现承载资源池化并提供统一调度能力，同时结合各层提供不同的数据交换能力与资源调控粒度，实现跨层资源间的协同规划及高效适配。通过发挥 TDM 层与

光层的刚性通道能力，满足带宽定制化与路由确定性要求，通过在分组层引入确定性调度机制，解决分组与转发时间的精准映射问题，实现同一接口内的各业务带宽、时延、抖动、丢包的定制化。

四、 关键技术创新与突破

4.1 异构混训

4.1.1 基于算力特征的模型分层拆解

为解决大模型在异构 GPU 混训时，因不同 GPU 算力特征存在差异而导致的混训同步难、训练算效低等问题，设计了一种基于算力特征的模型分层拆分方法。该方法首先构建起异构 GPU 算力特征与大模型架构层算力需求的多维评估体系，突破传统 FLOPS 单一指标的局限，从计算能力、显存特性、通信带宽等多个维度建立量化评估模型；同时，通过对大模型各层计算密度、内存访问模式等特征的量化分析，形成层计算特性，为后续的模型分层拆分提供精准依据。

（1） 异构 GPU 算力量化评估体系构建

针对异构 GPU 算力评估的复杂性，设计了一个包含计算能力、存储特性和通信能力三维度的评估框架。在计算能力维度，通过测量 FP16/INT8 混合精度下的浮点运算性能和张量计算吞吐量，来量化 GPU 的核心计算能力；存储特性维度则聚焦显存容量、显存带宽以及显存访问延迟等指标，以全面反映 GPU 的存储性能；通信能力维

度着重评估 GPU 间 NVLink 或 InfiniBand 等互联技术的通信带宽和延迟情况。可根据实际应用场景进行动态调整，以平衡不同维度对算力评估的权重，从而更精准地评估异构 GPU 的实际可用算力。

（2） Transformer 架构层特性分析

对大模型中广泛采用的 Transformer 架构进行深入研究，将其核心层划分为注意力层、前馈神经网络（FFN）层和层归一化等类型。针对注意力层，重点量化其多头注意力机制下的计算密度，包括矩阵乘法与累加操作的次数，以及因序列长度增长而带来的内存访问强度变化；FFN 层则侧重于分析其在不同激活函数下的计算复杂度，以及数据在多层感知机中流动时的内存读写模式；层归一化部分，详细研究其在不同数据规模下的计算开销，以及与其他层交互时的通信需求。通过对这些关键特征的量化分析，能够更清晰地把握 Transformer 架构各层在计算、存储和通信方面的特性，为后续的调度优化提供坚实的数据基础。

（3） 异构感知分层拆解算法设计

为充分发挥异构 GPU 集群的性能优势，设计了异构感知分层拆解算法。通过构建层特性矩阵和层间通信代价矩阵，将 Transformer 架构各层的计算、存储和通信特性，以及层与层之间的数据传输开销进行数字化建模。然后，利用动态规划算法求解初始分配方案，该方案以最小化整体计算时间和通信开销为目标，初步确定各层在不同 GPU 设备上的部署策略。

在此基础上，以计算时间、通信时间和能耗惩罚为奖励函数，通

过不断的试错与学习，对初始分配方案进行迭代优化。算法能够实现自适应层融合，将计算量较小的层进行合并处理，减少不必要的通信开销；通过通信感知调度，根据 GPU 间的实时通信状态动态调整数据传输路径；借助弹性流水线技术，平衡各 GPU 设备的负载，从而在计算时间、通信时间和能耗惩罚之间找到最优平衡点，显著提升大模型跨域训练的效率和资源利用率。

4.1.2 自适应训练任务运行时配置

为动态适配异构跨域 GPU 资源的大模型混训，需根据混训方案动态生成适配异构 GPU 的镜像与配置文件。为此，提出异构跨域 GPU 的大模型混训动态镜像与配置文件匹配方法，能够动态地匹配合适的镜像和配置文件，提高模型训练的效率和资源利用率，同时确保任务在不同的计算环境中能够稳定、高效地运行。主要模块与流程如图 9 所示。

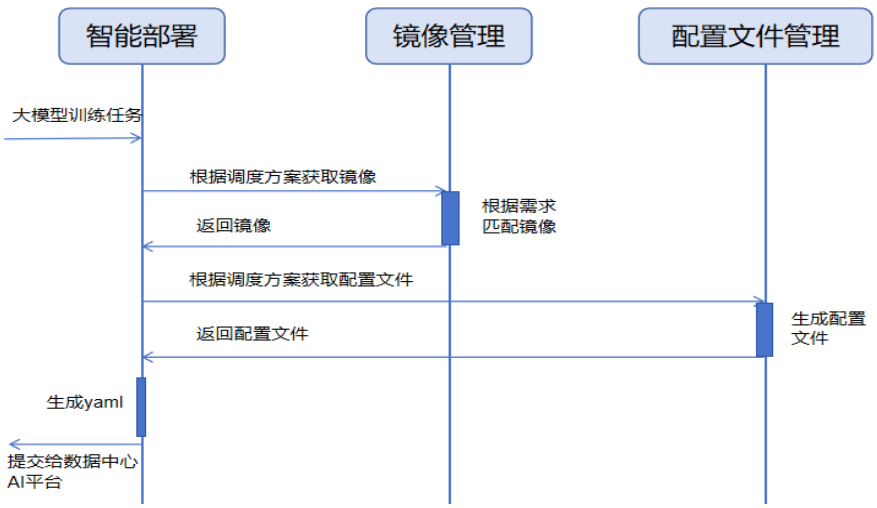

图 9　动态镜像和配置生成工作流程

当大模型混训任务发起后，协同调度模块解析任务的计算与数据需求，生成蓝图传递给智能部署模块。智能部署模块依蓝图向镜像管理模块查询，后者从异构仓库筛选匹配镜像并反馈信息；同时，智能部署模块向配置文件管理模块请求配置，该模块调用模板库生成配置文件返回。最终，智能部署模块整合镜像与配置，生成各集群作业描述文件，提交至对应智算集群执行。

（1） 智能部署模块

智能部署模块负责接收大模型混训任务，根据任务的计算需求、数据分布以及不同地域 GPU 的实时状态（包括算力、显存、负载等），采用智能调度算法将任务合理分配到合适的 GPU 资源上。该模块确定了调度的资源以后向镜像管理模块查询各训练实例上使用的具体镜像，向配置文件管理模块查询训练实例上使用的具体配置，然后根据镜像和配置生成作业描述文件，调度到具体的 AI 平台。

（2） 镜像管理模块

镜像管理模块维护动态匹配镜像仓库，存储各类训练镜像，以名称和标签标识镜像特征与能力。用户提交训练代码及信息后，镜像生成模块据此打包镜像，并将其名称、标签存入存储模块。智能调度模块依据任务调度结果，向镜像管理模块查询匹配镜像。此外，该模块支持动态更新与版本管理，可适配 GPU 硬件及软件升级需求。

（3） 配置文件管理模块

配置文件管理模块基于调度模块生成的蓝图中训练任务的特点，根据配置文件模板库生成具体的训练作业的配置参数。模板中包含大

模型训练参数、资源参数、存储挂载参数、网络通信库、网络通信参数。该模块能够根据实际调度结果，从模板库中选取合适的模板，并能够根据大模型训练任务的具体情况调整其中的参数，最后动态生成针对特定任务和 GPU 的配置文件。

- **训练参数**

配置文件管理模块存储所有镜像支持的训练参数及传递方式（如启动参数、环境变量等）。生成配置文件时，按对应传递方式构建。

训练参数传递方式依场景而定：选择“基础镜像+系统代码”，由代码能力确定；选择“基础镜像+自有代码”，需先将自有代码注册至配置文件管理并指定传递方式；选“自定义镜像+代码内置”，则从镜像信息获取。最终，结合训练需求与传递方式，生成含学习率、批量大小等关键参数的训练作业配置。

- **资源参数**

资源参数包括 GPU 型号、GPU 数量、网卡型号、网卡数量、CPU 型号、CPU 大小、内存大小等。配置文件管理模块根据蓝图需求生成具体的资源参数。

- **存储挂载参数**

蓝图中包含了实际数据存储位置、checkpoint 存储位置；当用户选择了基础镜像，训练代码通过存储系统提供时的代码实际存储位置；系统提供的容器启动脚本实际存储位置。配置文件管理模块将这些存储位置映射到容器中的蓝图指定位置。

- **网络通信库**

配置文件管理模块中存储了每个镜像支持那些通信库、选择通信库的方式。若蓝图中指定了通信库，首先判断镜像是否支持通过某种选择通信库，支持的通信库中是否包含指定通信库，如果支持则直接通过指定方式生成配置文件（启动参数、环境变量等）；如果不支持则根据其他方式指定通信库，如替换.so 文件。若蓝图中没有指定通信库，那么配置文件管理模块需要判断大模型训练使用的网络特性，比如是否需要跨域，如果跨域则使用支持跨域优化的数据库。

- **网络通信参数**

确定网络通信库以后，首先判断蓝图中是否有用户指定的网络优化，如果有，则根据用户的网络优化生成指定的网络通信参数；如果没有则根据收集的网络情况生成网络通信参数：若没有跨域，则使用默认网络通信参数；若需要跨域通信，且广域网可以提供确定性网络传输，则设置缓冲区、消息大小为较大的值（具体根据 RTT）；如果需要跨域通信，但广域网不能提供确定性网络传输，则设置为使用指定的网络传输协议。

4.2 异地同训

4.2.1 计算通信重叠的流水线并行

为解决跨域长距网络环境引发的串行流水线等待问题，设计了一种计算与通信重叠的流水线并行加速技术，如图 10 所示。在该技术方案中，流水线内的大部分 GPU 通信可与 GPU 计算重叠执行，可极

大降低对 GPU 服务器之间网络带宽、延迟的要求。当在低带宽、高延迟的网络环境下采用这种重叠执行方式时，可减少 GPU 计算的等待时间，从而提升大模型训练效率以及 GPU 算力利用率。

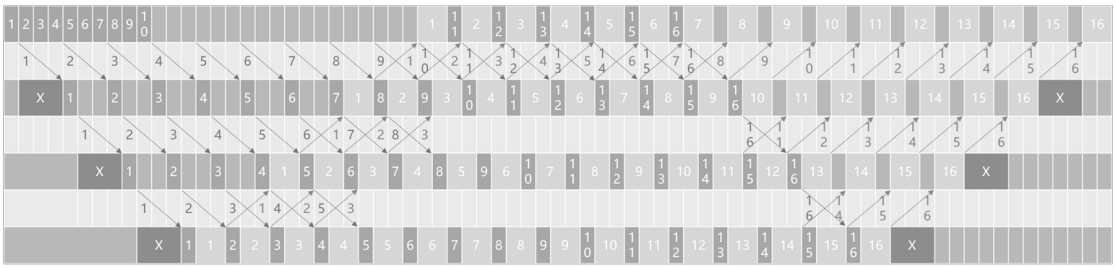

图 10　计算与通信重叠的流水线

该技术方案有两个技术思路：一是在流水线热身阶段，尽可能多得进行前向传播计算，也就是尽可能增加流水线热身阶段微批次数量；二是使前、后传播的 GPU 计算与流水线内的 GPU 通信重叠执行。GPU 计算与 GPU 通信重叠执行的实现方式是在每次前、后传播的 GPU 计算开始前，启动一个或多个非阻塞 GPU 通信（即接收、发送前、后传播计算结果）过程。

流水线中每一个 Worker 的运行过程包含以下步骤：根据提前约定的策略确定流水线热身、稳定、冷却阶段的微批次数量；以计算与通信重叠的方式执行流水线热身阶段的计算与通信操作；如果流水线稳定阶段的微批次数量不为 0，以计算与通信重叠的方式执行流水线稳定阶段的计算与通信操作；如果流水线稳定冷却的微批次数量不为 0，以计算与通信重叠的方式执行流水线冷却阶段的计算与通信操作。

（1）流水线热身阶段

流水线热身阶段包含以下计算与通信操作。首先进行初始化设置，

令 N 和 M 的初始值均为 1，随后接收第 N 个微批次的前向传播计算结果。对于热身阶段的每个微批次，需按以下步骤执行操作：若当前微批次不是当前训练迭代的最后一个，启动第 N+1 个微批次前向传播计算结果的非阻塞接收过程；接着执行第 N 个微批次的前向传播计算。若当前微批次是流水线热身阶段的最后一个，需同时启动第 N 个微批次前向传播计算结果的非阻塞发送过程，以及第 M 个微批次后向传播计算结果的非阻塞接收过程，完成后转入流水线稳定阶段；若不是，则直接启动第 N 个微批次前向传播计算结果的非阻塞发送过程。之后，等待第 N+1 个微批次前向传播计算结果的非阻塞接收完成，并将 N 的值更新为 N+1。

（2）流水线稳定阶段

对于稳定阶段的每个微批次。首先等待第 N+1 个微批次前向传播计算结果的非阻塞接收完成，并将 N 的值更新为 N+1。随后根据当前微批次所处位置执行对应操作：若为流水线热身阶段的第一个微批次，启动第 N+1 个微批次前向传播计算结果的非阻塞接收过程；若为流水线热身阶段的最后一个微批次，启动第 M-1 个微批次后向传播计算结果的非阻塞发送过程；其他情况则同时启动第 N+1 个微批次前向传播计算结果的非阻塞接收过程，以及第 M-1 个微批次后向传播计算结果的非阻塞发送过程。

接着执行第 N 个微批次的前向传播计算，等待第 M 个微批次后向传播计算结果的非阻塞接收完成后，启动第 N 个微批次前向传播计算结果的非阻塞发送过程，以及第 M+1 个微批次后向传播计算结

果的非阻塞接收过程。随后执行第 M 个微批次的后向传播计算，并将 M 的值更新为 M+1。

最后根据当前微批次状态进行收尾处理：若为稳定阶段的最后一个微批次且流水线冷却阶段需处理的微批次数量不为 0，启动第 M-1 个微批次后向传播计算结果的非阻塞发送过程并转入冷却阶段；若为稳定阶段的最后一个微批次且冷却阶段无需处理微批次，则发送第 M-1 个微批次的后向传播计算结果并停止流水线运行。

（3）流水线冷却阶段

对于冷却阶段的每个微批次，首先等待 M 个微批次后向传播计算结果的非阻塞接收完成。若当前微批次并非冷却阶段的最后一个，启动第 M+1 个微批次后向传播计算结果的非阻塞接收过程。接着执行第 M 个微批次的后向传播计算。

若当前微批次是冷却阶段的最后一个，发送第 M 个微批次的后向传播计算结果并停止流水线运行；若不是，则启动第 M 个微批次后向传播计算结果的非阻塞发送过程。最后将 M 的值更新为 M+1。

4.2.2 高效非阻塞 GPU 通信技术

为解决 GPU 同时运行计算与通信任务时的底层资源竞争问题，设计了一种非阻塞 GPU 通信方法。该方法通过内存作为数据中转枢纽，先将显存中的数据迁移至内存，再基于内存中的数据执行通信操作，最后将通信完成后的数据从内存迁回显存。由于基于内存数据的通信操作仅依赖 CPU 而不占用 GPU 资源，这一机制能大幅减少 GPU

通信与计算在同一 GPU 卡上的资源竞争，从而有效避免因资源争抢导致的通信时间增加。

该技术方案包含两个核心思路：其一，确立内存的中介地位，构建 "显存→内存→通信→内存→显存" 的数据传输链路，通过内存缓冲实现计算与通信的数据隔离；其二，采用独立的通信进程或线程专门负责基于内存数据的通信操作，使其与 GPU 计算任务在执行层面完全分离，确保通信过程不会干扰 GPU 计算的正常运行节奏。

（1）专用通信进程方案

在 GPU 计算任务所在进程（即 GPU 计算进程）之外，增加专用通信进程，用于执行基于内存数据的通信操作。GPU 计算进程与通信进程之间通过共享内存传输数据。非阻塞 GPU 通信流程如图 11 所示。

GPU 计算进程根据通信操作类型，依次将显存数据传输至非分页内存，再复制到共享内存，随后发送启动信号并启动计算任务。计算完成后，发送等待信号并等待响应，最后将通信后的数据依次复制回非分页内存和显存。

专用通信进程接收到启动信号后执行通信操作，收到等待信号时，待通信结束后向 GPU 计算进程发送响应。

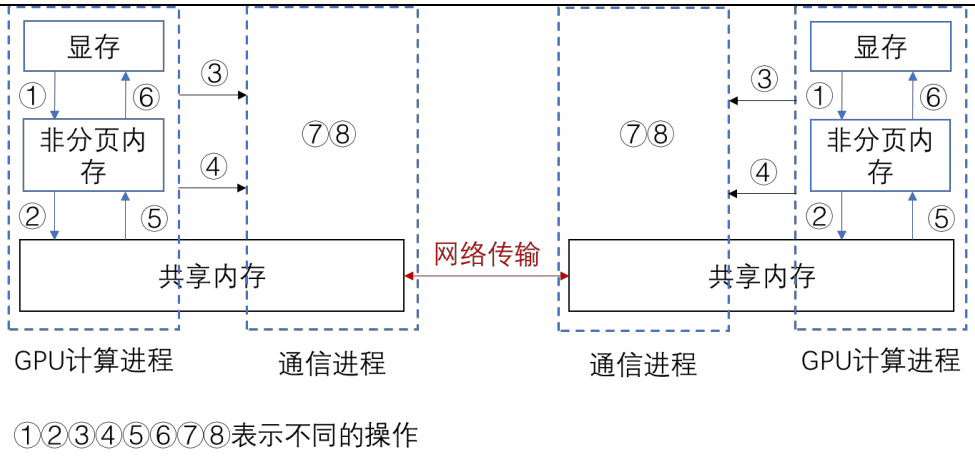

图 11　基于专用通信进程的非阻塞 GPU 通信

（2）专用通信线程方案

在 GPU 计算任务所在线程（即 GPU 计算线程）之外，增加专用通信线程，用于执行基于内存数据的通信操作。非阻塞 GPU 通信流程如图 12 所示。

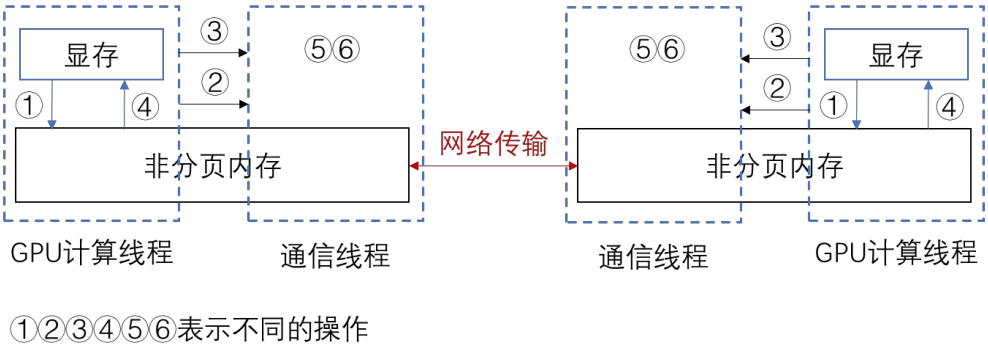

图 12　基于专用通信线程的非阻塞 GPU 通信

GPU 计算线程和专用通信线程根据通信操作类型协同工作。GPU 计算线程先将显存数据传至非分页内存，发通信启动信号并启动计算任务；计算完成后发等待信号，待通信结束将数据传回显存。专用通信线程收到启动信号执行通信，收到等待信号则完成通信后响应。

以两节点点对点 Send/Recv 通信为例：发送方节点 1 的 GPU 计算进程，先将显存数据转至非分页内存并复制到共享内存，发 Send 启动信号并开始计算，计算完成后等待 Send 操作结束。接收方节点 2 的 GPU 计算进程，先发 Recv 启动信号并启动计算，计算完成后等待 Recv 结束，最后将通信后的数据依次存入非分页内存和显存。

4.2.3 面向大模型跨域训练的算网协同调度

为降低大模型跨域训练对广域网带宽需求，减少用户跨越训练大模型时需要付出的额外成本，需要对大模型训练作业的通信需求、网资源拓扑进行建模，通过算网协同调度选择成本最低的算力集群组合以及与之适配的大模型训练作业拆分方案。

（1）大模型训练作业需求建模

在大模型训练的 3D 并行模式中，通信流量特征因策略而异。混合 3D 并行时，不同策略的流量叠加会引发带宽竞争，加剧网络负载不均衡。这些特性对网络与调度提出特殊要求，是影响训练效率的关键，其中：数据并行 DP 对延迟敏感，千卡级集群需数百 Gbps 带宽；张量并行 TP 需微秒级响应，每层参数切分交换可达数百 GB；流水线并行 PP 单次微批次传输约数 GB，数十 Gbps 带宽即可满足。

在跨域训练场景下流水线并行的流量特征与广域网的高延迟、有限带宽特性具有天然适配性。通过将 PP 合理分布于不同域，可有效利用广域网的异构资源，降低全局同步开销，为大模型训练提供更灵活的分布式扩展路径。

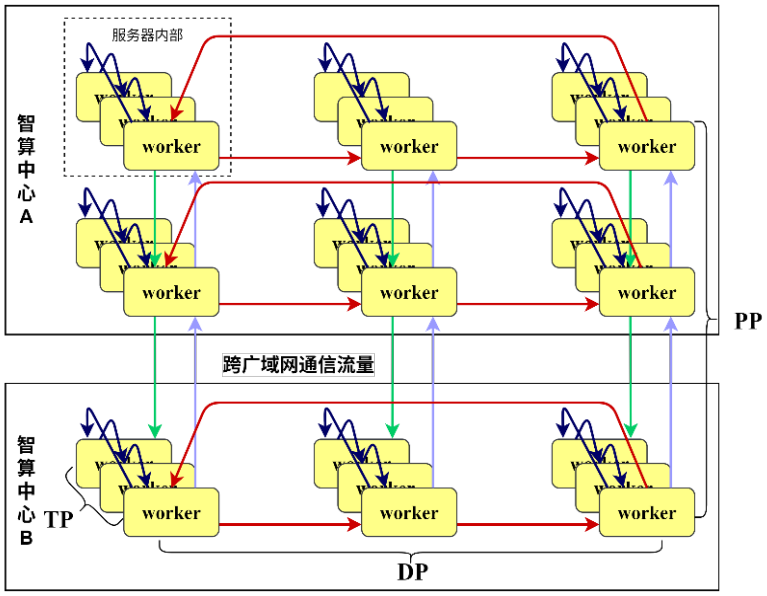

图 13 大模型跨域训练部署示意图

如图 13 所示，根据大模型不同并行模式的通信特征，本技术方案将张量并行约束在单个服务器内部，将数据并行约束在单智算中心内部，将大模型流水线并行的通信流量放在广域网上，由此对大模型训练的 GPU、广域网需求进行建模。

在大模型训练中，张量并行限于单服务器内通信，数据并行也仅在同一智算中心内完成。因此，大模型跨广域网训练时，主要需考虑流水线并行的网络流量。其通信量源自前向传播的激活值传输与反向传播的梯度传递，具体数值由模型结构、阶段划分和微批次设置决定。简单来讲，单次前向或反向传播的通信量与序列长度、隐藏层维度及单个数据存储大小相关，且每级流水线通信需求相近。网络要求可用三元组表示为 <Bandwidth，delay，jitter>，故跨广域网训练对广域网络的总需求为 d 条流水线的网络需求集合，即 <d×Bandwidth，Delay，Jitter>。

（2）算网资源拓扑建模

为实现大模型训练作业的跨广域网调度与部署，需对广域网环境的算网资源进行建模。大模型跨广域训练主要的关注点在于智算中心的各类 GPU 数量与智算中心间的确定性广域网的能力。为此，在算网资源方面，需对智算中心各类 GPU 资源量与各智算中心的网络能力进行建模。如下图所示，将智算中心抽象为节点，将智算中心间的互联网络抽象为节点间的连线。

智算中心内部部署多种异构 GPU，需对各类型算力资源进行细粒度计量。由于大模型训练业务对 GPU 算力密度和通信效率具有严苛要求，GPU 节点内并行计算能力、节点间数据传输效率，以及跨数据中心协同性能，均受底层拓扑结构显著影响。因此，资源计量需精确至节点层级，即准确统计各节点 GPU 卡数量，而非简单汇总整个智算中心的 GPU 资源总量，具体统计示例如表 1 所示：

表 1　GPU 资源统计表

GPU 型号	单节点卡数	节点数量
英伟达 A100	8	16
昇腾 910B	8	10

智算中心间的网络链路连接关系采用<智算中心 1，智算中心 2>二元组表示，用于刻画任意两个智算中心之间的网络联通状态。网络性能评估采用<Bandwidth, Delay, Jitter>三元组模型。

（3）面向大模型训练作业的跨广域协同调度策略

该协同调度策略旨在根据大模型训练 GPU 需求与 GPU 资源节点拓扑，求解大模型跨广域训练的部署方案，即大模型训练 Worker 与 GPU 资源节点的绑定及其 GPU 资源、存储资源的分配，大模型训练跨域通信与广域网的网络链路的绑定与网络资源的分配。具体协同调度策略由协同调度、算力调度、存储调度、网络调度以及其三者的协同调度四部分构成。

- **协同调度策略**

大模型训练作业的协同调度流程为协同调度驱动算力调度、网络调度、存储调度协作的过程。首先检查等待处理的作业队列，若队列空无任务，便等待下一个调度周期；若存在待处理任务，则依照队列的排队规则选取一个大模型训练作业。选中作业后，先对其进行预处理：结合作业描述与预设的需求建模规则，明确每级流水线的 GPU 资源需求。接下来，初始化一个空的待选方案集合，进入算力调度阶段。依据特定的算力调度策略，筛选出满足作业算力需求的资源节点，形成初步的调度方案集合，并标记算力调度完成。之后更新待选方案集合，检查集合是否为空：若为空，说明暂无符合条件的方案，返回初始步骤等待；若不为空，则判断存储资源是否已分配。若存储资源未分配，进入存储调度阶段，按存储需求筛选方案，剔除不满足条件的选项，标记存储调度完成后再次更新待选集合。若存储资源已处理完毕，则检查网络资源分配状态。若网络资源未分配，进入网络调度阶段，依据网络需求筛选方案，移除不符合要求的选项，标记网络调度完成后更新待选集合。当算力、存储、网络调度均完成后，从待选

方案集合中输出最终的调度方案。

- **算力调度策略**

算力调度核心思想是以大模型流水线的每一级为 GPU 算力的分配单元，确保大模型跨域训练任务按流水线级数进行切分。

初始 GPU 需求筛选范围。待分配 GPU 的大模型流水线级数编号即为 i 到 j，其中 $0 \le i \le j < p$，$i \in N$，$j \in N$， p 为大模型流水线并行的总级数，i 初始值为 0，j 初始值为 p-1。初始化可选的智算资源节点集合。

筛选满足 GPU 需求的智算资源节点。统计从 i 到 j 的所有流水线级数的各类资源需求量。遍历 i 到 j 的各类 GPU 资源需求，对同卡数同类型的 GPU 节点数加和。遍历各智算资源节点集合，若某智算资源节点的各类 GPU 资源节点满足从 i 到 j 的所有流水线级数的各类资源需求量,则将该智算资源加入待优选节点集合。

判断 GPU 的节点集合是否为空。若空，则需调整 GPU 筛选范围，则 j 更新为 j-1。存在满足 GPU 需求的智算节点，标记 GPU 需求与节点集合的对应关系。遍历满足 GPU 需求的智算节点集，将这些智算节点添加到作业筛选范围内所在流水线级数的训练任务可选 GPU 智算节点集合。

判断所有 GPU 需求是否满足。判断从 0 到 p-1 所有级数的 GPU 需求均分配智算资源。若还存在未分配资源节点，更新 GPU 需求筛选范围，更新 i 为 j+1，j 更新为 p-1。根据每级流水线与智算节点的分配关系，生成满足 GPU 需求的调度方案。若 i>j，则表示筛选范围

为空，即算力调度失败，结束。反之，表示筛选范围不为空，可继续算力调度。

- **存储调度策略**

存储调度的目标是从待选集合中筛选出满足大模型训练作业存储需求的方案。

从待选方案集合中筛选出尚未完成存储资源匹配的方案，作为待处理对象。随后遍历各方案的流水线层级，依据算力调度所确立的流水线 Stagei 与智算节点 Nodek 的映射关系，对每个 Nodek 的存储资源需求进行累加计算，从而获取整体存储需求总量。

对各智算节点 Nodek 的存储资源容量与大模型训练作业累计存储需求进行对比分析。若某节点存储资源无法满足需求，则将该方案从待选集合中剔除；若满足需求，则标记该方案为“存储需求匹配完成”，并进入下一阶段处理。

核查待选集合中是否仍存在未完成存储资源匹配的方案。若存在，则返回流程初始步骤，继续处理下一个未匹配方案；若全部方案均已完成存储资源匹配，则标志着整个存储调度流程执行完毕。

- **网络调度策略**

网络调度的目标是从待选方案集中识别大模型跨域数据传输与通信的需求，并判断网络资源是否满足其需求，进一步从待选方案集中过滤掉不满足网络需求的方案。

在待选方案集合中实施筛选操作，甄别出尚未完成网络资源匹配的方案，并将其确定为待处理对象。其次，针对各方案的流水线层级

进行系统性遍历，依据算力调度构建的流水线与智算节点的映射关系，对所有流水线判断。若相邻的两个流水线级对应同一智算节点，表明该两级间仅存在智算中心内部通信；若相邻两级对应不同智算节点，则意味着存在跨广域通信需求，即涉及跨广域网的网络资源需求。

对广域网承载能力与相邻流水线级间的通信流量需求开展严格的对比分析。若该广域网无法达到需求阈值，则将相应方案从待选集合中移除；若满足需求条件，则将该方案标记为“网络需求匹配完成”，并进入下一处理阶段。

对剩余待选方案集合进行核查，判断是否仍存在未完成网络资源匹配的方案。若存在此类方案，则返回流程初始阶段，继续处理下一个未匹配方案；若所有方案均已完成网络资源匹配，则视为整个网络调度流程执行完毕。

4.2.4 跨域训练 RDMA 加速网关

为了解决 RDMA 在广域网上效率低下的问题，RDMA 网关通过拥塞控制和精细化的报文处理与会话管理机制加速 RDMA 通信，实现大带宽低时延的 RDMA 操作。

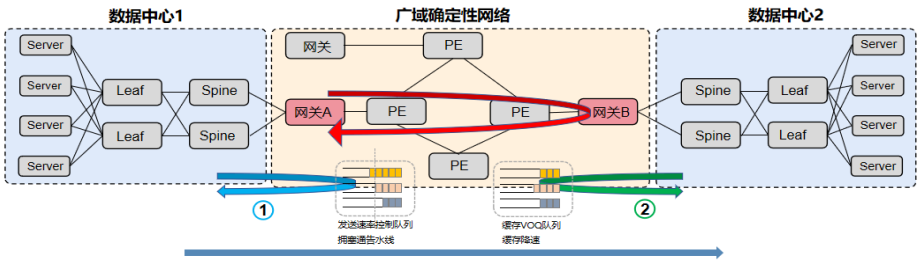

图 14 长距 RDMA 拥塞控制

长距 RDMA 拥塞控制如图 14 所示，主要流程如下：**近源拥塞控制。**源数据中心内，源服务器至源网关的路径上出现拥塞时，采用预设拥塞控制算法生成拥塞通告报文，并发送至源服务器，源服务器根据拥塞通告报文调整发送速率；**近目的端调整发送速率。**目的数据中心内，目的网关发送至目的服务器的数据包形成拥塞时，目的服务器会发出拥塞通告报文至目的网关，目的网关根据到达的拥塞通告报文调整对应流量的发送速率和队列缓存；**端到端控制。**目的网关每隔预设时间段统计队列的发送速率，根据队列发送速率生成长距拥塞通告报文，并发送至源网关；源网关根据长距拥塞通告报文中记录的发送速率和当前队列速率进行比较，以调整转发速率。

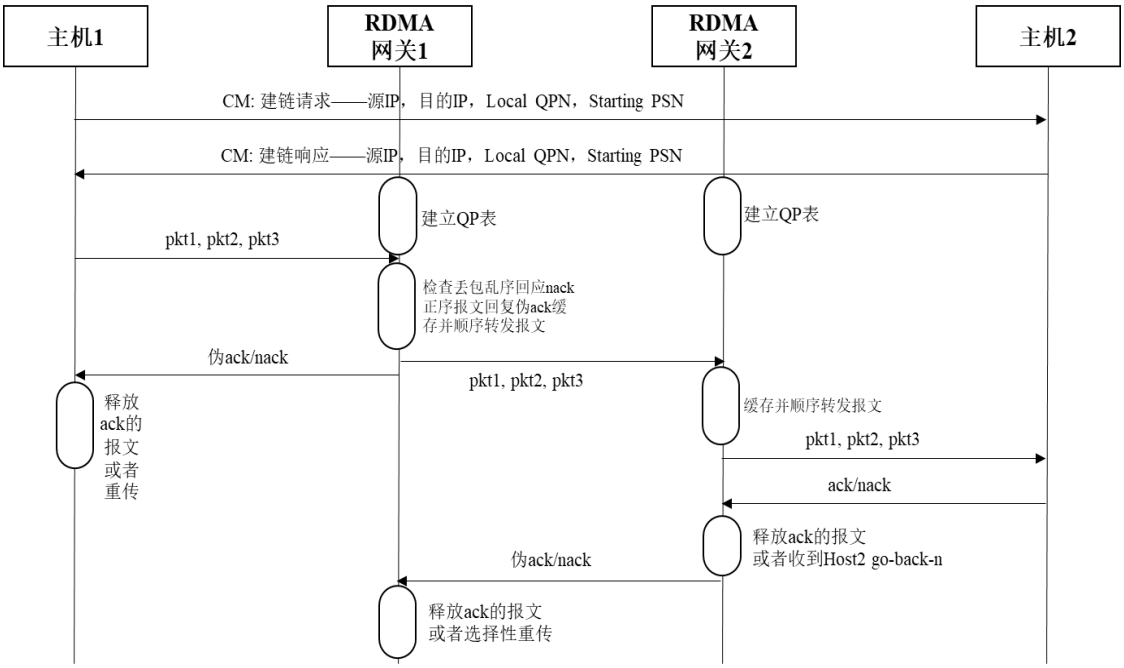

图 15　长距 RDMA 工作流程

RDMA 网关精细化报文处理与会话管理机制的工作流程如图 15 所示，具体流程如下：

（1）QP 业务流表建立

当 RDMA 网关收到来自源端的 RDMA 报文时，首先基于 CM 会话交互过程进行会话初始化。在会话建立阶段，网关会为每个独立的通信流创建专属的 QP 业务流表，每个流表绑定独立的队列缓存空间。这种一对一的流表 - 缓存映射机制，可实现不同通信流的隔离处理，避免跨流数据干扰，同时为后续的丢包检测、重传控制提供精准的粒度支持。

（2）数据报文处理逻辑

报文接收与状态判断。网关收到 RDMA 数据报文后，通过序列号校验检测丢包或乱序。异常时发 NACK 触发上游选择性重传，仅补传缺失片段；正常则发伪 ACK 确认接收，并缓存报文等待下游最终确认，避免上游超时重传。**确认报文处理**。收到下游 ACK 即释放缓存，收到 NACK 则重发对应报文，确保数据完整传输。

（3）重传与超时控制机制

重传次数限制。网关支持重传次数配置，当报文重传达设定阈值，主动释放缓存并标记传输异常，触发重试重启数据传输，避免资源阻塞。**超时重传策略**。网关为报文设可配置超时定时器（时长大于链路 RTT），超时尚未收到下游 ACK 或 NACK，自动重传报文，弥补链路瞬时故障，保障传输可靠。

通过上述机制，RDMA 网关在异属异构智算资源的跨域通信中，实现协议透明传输，优化长距链路性能。

4.2.5 网络状态感知的负载均衡

在大模型训练的负载分担中，一般使用基于五元组的方式逐流负载分担，或者逐包负载分担。这都是基于域内的数据并行和张量并行数据量很大，需要细粒度的拆分，才能够将大流量分流到不同的链路上。然而，在大模型跨域训练过程中，PP 的 RDMA 流量一般会在网关上做加速处理，这要求 RDMA 流量的往返路径都经过同样的网关；当大模型训练跨域 PP 较多、流量较大时，又希望流量能够在多个网关间负载均衡。传统基于 HASH 的负载均衡无法满足这种要求。使用多网关负载均衡情况下，大模型跨域训练流量单个 QP 对的流量如图 16　跨域训练流量示例所示。

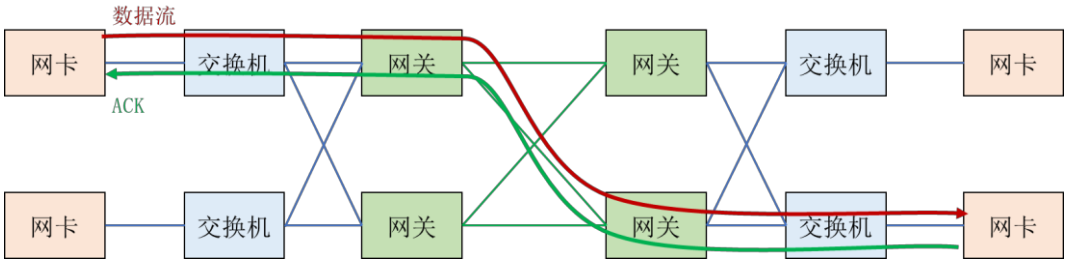

图 16　跨域训练流量示例

为此，设计大模型跨域训练 RDMA 流量负载均衡方案，适配高流量跨域通信场景。大模型训练常用 RoCEv2 协议，以 UDP/IP 封装 InfiniBand 语义，固定目的端口 4791 标识 RDMA 流量，源端口动态分配区分会话，网络设备仅解析 L4 头部。RoCEv2 中，数据靠 QP 对传输，QP 绑定唯一五元组，路径选择粒度小于五元组会致数据包失序，引发效率下降：

- 序列号校验失效：数据包携带序列号，乱序触发接收端 NACK，导致大量重传降速。即便支持 SACK，也需暂存乱序包等待重组，受缓冲区限制影响效率。
- 流控误判：RoCEv2 依赖 ECN 反馈拥塞，乱序易使接收端混淆丢包原因，错误触发发送端流量暂停或降速，造成带宽波动。
- 硬件资源浪费：主流 RoCEv2 网卡接收队列采用 FIFO 结构，乱序包缓存占用额外资源，队列满时后续包会被硬件丢弃。

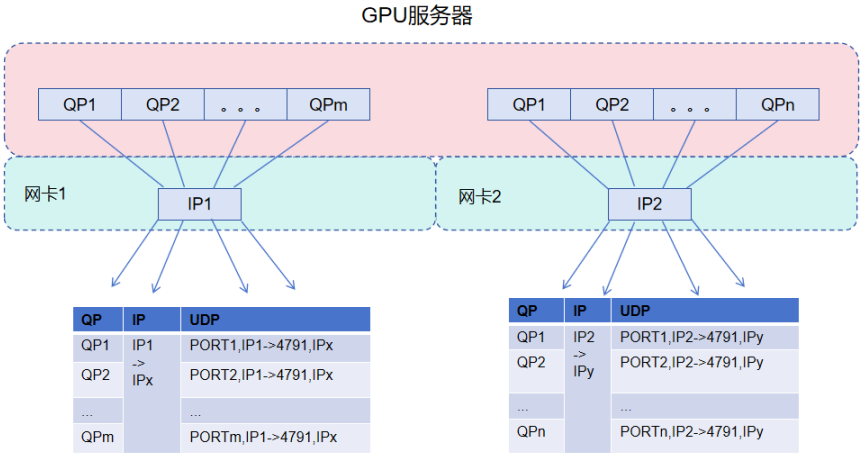

图 17　QP 与流量间的关系图

对于 RoCEv2 来说，负载分担的粒度不宜小于五元组，才能在效率上达到较好的状态。具体到大模型训练时，每个网卡具有一个 IP，一对网卡之间一般使用多个 QP 发送和接收流量。如图 17　QP 与流量间的关系图所示为一个具有两个网卡的 GPU 服务器，地址分别为 IP1 和 IP2，每个网卡上各使用了 m 和 n 个 QP，每个 QP 对应唯一的五元组。

网卡发出的流量如图 18 网卡产生流量与 QP 关系所示，基于大模型训练流量的上述特征，通过不同的服务器架构和大模型训练模式等提取不同流量粒度进行负载分担。

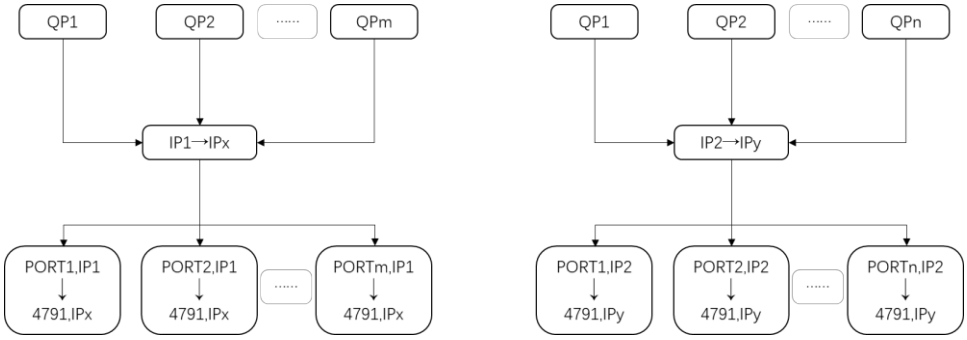

图 18 网卡产生流量与 QP 关系

控制器依据 GPU 服务器架构与大模型训练模式，确定负载分担流量粒度；再根据模型特点和部署位置，明确流量通信需求（带宽、时延等），向确定性网络控制器请求开通隧道。隧道开通策略为：优先请求各 PE 上的同路径隧道，若全部失败则转而请求多路径隧道，均失败则返回失败。隧道开通后，通过可插拔算法计算权重选择网关，并将负载分担策略下发至网关，VxLAN 路由下发至网卡。控制器计算并下发完表项后，各转发部件的具体转发过程如图 19 所示：

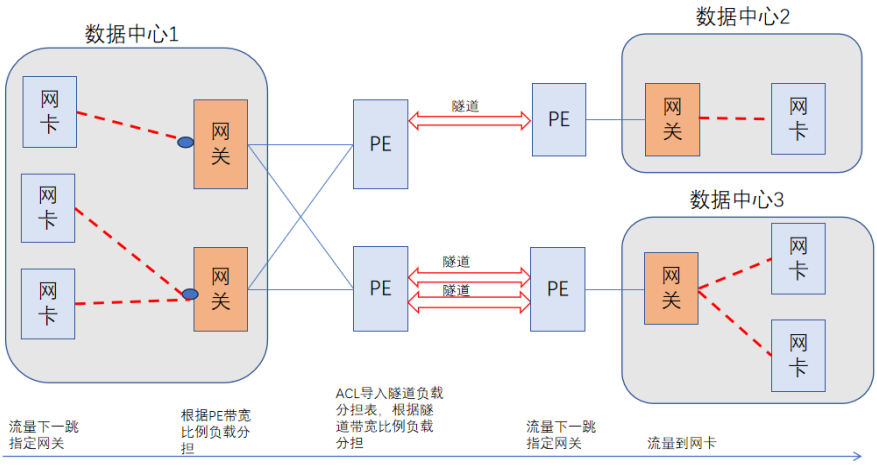

图 19 端到端转发示意图

发送端网卡根据流量特性指定下一跳网关，完成 VxLAN 封装后经 Underlay 网络转发。发送端网关解封装后，依据目的网卡负载分担策略（按 PE 隧道带宽比例）将流量发送至 PE。PE 处理时，单隧道单路径模式直接导入指定隧道，单隧道多路径模式则按路径带宽比例负载分担。接收端 PE 按流量目的特性将流量发送至网关，网关完成 VxLAN 封装后送达目的网卡。

4.2.6 广域 RDMA 通信代理技术

因广域网时延太大、丢包率较高、抖动明显，传统 RDMA 在跨广域网通信时效率很低，为此设计了一种广域网环境下分段 RDMA 通信链路建立方法，将 RDMA 连接分成三段：源服务器的 RDMA 通信连接到源代理服务器终结，源代理服务器通过更适应广域的通信协议将数据发送到目的代理服务器，目的代理服务器将数据转换为 RDMA 通信发送给目的服务器，如图 20 所示。

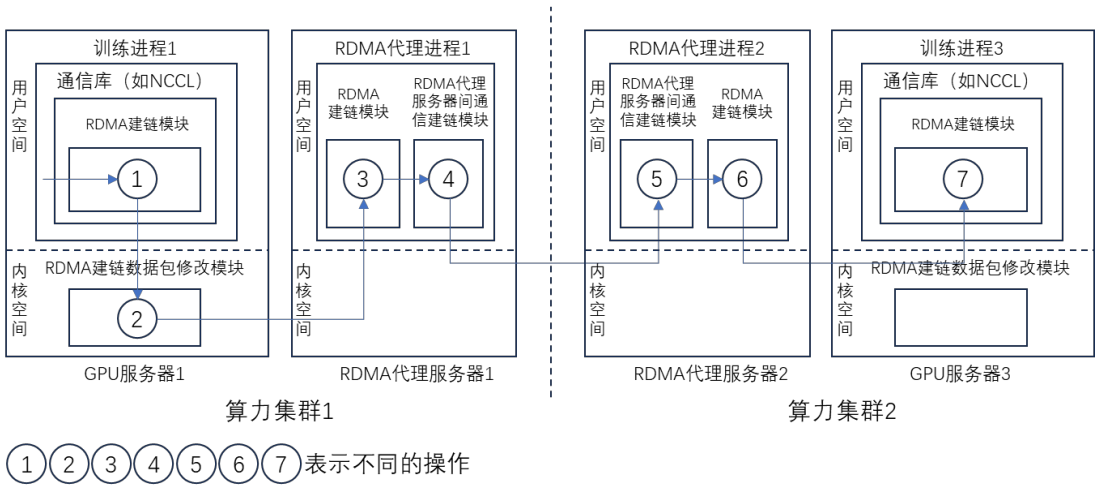

图 20 分段 RDMA 通信链路建立方法示意图

RDMA（远程直接内存访问）控制面的作用是为通信两端建立RDMA 通信链路。RDMA 数据面的作用是在建立好的 RDMA 通信链路上传输业务数据。RDMA 数据面的通信必须依赖 RDMA 网卡来完成。但 RDMA 控制面的通信不依赖 RDMA 网卡，可以采用任何合适的通信技术（如 TCP、UDP、QUIC）。所谓 RDMA 通信建链，就是为通信两端上的 QP（Queue Pair）建立对应关系。在 RDMA 通信建链过程中，通信两端会交换要建立对应关系的 QP 的相关信息，包括但不限于 QP 编号、数据包序列号、全局 ID。

使用多算力跨域协同训练的情况，此时需要把训练进程 1 和训练进程 3 分别放在两个异地算力集群运行。但是对于训练进程 1 和训练进程 3 来说，他们并不知道对方和自己不在一个算力集群。因此，训练进程 1 和训练进程 3 依然按照普通 RDMA 建链方法的逻辑去执行与 RDMA 建链相关的各种操作。按照普通 RDMA 建链流程，训练进程 1 仍然作为普通 RDMA 通信建链的发起方，将普通 RDMA 通信建链请求发送给训练进程 3，而训练进程 3 仍然作为普通 RDMA 通信建链的接受方，等待由训练进程 1 发送的普通 RDMA 通信建链请求。这样的话，就无法建立分段 RDMA 通信链路。

跨广域网的 RDMA 分段 建链流程可拆解为七个核心操作：

（1） 初始建链请求

在分布式训练场景下，训练进程 1 依据通信库定义的标准 RDMA 建链协议规范，向训练进程 3 发起建链请求。此时数据包源 IP 配置为 192.168.1.1，目的 IP 设为 192.168.1.3。由于传统直连方

式无法满足分段式 RDMA 通信架构需求，若保持数据包原始路由信息，将直接抵达训练进程 3 所在的 GPU 服务器 2，导致无法构建预期的分层通信链路，因此需执行后续处理流程。

（2） 数据包重定向

训练进程 1 所在的 GPU 服务器 1 依据预先设定的路由策略，对建链请求数据包进行深度报文解析与修改。具体操作包括：将数据包目的 IP 及端口重定向至 RDMA 代理进程 1 的控制面 IP 与服务端口，并在数据包扩展字段中嵌入原始建链目标（训练进程 3）的完整元数据信息。在复杂算力集群环境中，当存在多台 RDMA 代理服务器及多个代理进程实例时，需通过负载均衡或优先级调度等策略进行代理进程选择（本文暂不探讨具体算法实现）。每个代理进程均绑定至少一组唯一的控制面 IP 与服务端口，作为接收建链请求的标准接口。

（3） 代理进程建链

RDMA 代理进程 1 接收到符合通信库标准协议格式的建链请求后，严格遵循既定的 RDMA 握手协议流程，与训练进程 1 完成基础通信链路的建立。该过程包含多次往返的控制报文交互，以协商链路参数并完成身份验证。

（4） 集群间链路建立

链路标识确定：系统对操作 3 中建立的所有普通 RDMA 通信链路进行唯一标识符分配与管理，确保在多链路并行场景下，每个通信通道均可通过唯一标识进行精准识别与区分。

原始目标提取：通过解析数据包扩展字段，完整提取操作 2 中附加的原始建链目标元数据信息，包括但不限于目标进程地址、端口及相关配置参数。

- 目标端确定：依据预定义的集群间路由规则，结合提取的原始目标信息，采用启发式算法或确定性策略选定集群间通信链路的建链目标（如 RDMA 代理进程 2），并发起标准化的集群间链路建立请求（具体的代理进程选择机制超出本文研究范畴）。
- 信息传递：在集群间链路建立过程中，通过带外控制通道或已建立的低延迟链路，将普通 RDMA 链路标识及原始建链目标信息（例如训练进程 1 与 RDMA 代理进程 1 间的链路唯一标识、训练进程 3 的控制面 IP 及服务端口等关键参数）可靠传输至集群间链路目标端（RDMA 代理进程 2）。

（5） 完成集群间链路

RDMA 代理进程 2 接收到集群间通信链路建立请求后，基于预设的通信协议栈（可选用 RDMA、TCP、UDP、QUIC 等传输协议），执行完整的链路协商与建立流程，实现代理进程间的高效数据传输通道构建。

（6） 二次建链请求

RDMA 代理进程 2 根据接收到的原始建链目标信息及普通 RDMA 链路标识集合，按照标准 RDMA 建链协议规范，向最终目标节点（训练进程 3）发起二次建链请求。该请求包含完整的链路上下

文信息，确保目标节点能够准确识别并响应。

（7） 最终建链完成

训练进程 3 接收到来自 RDMA 代理进程 2 的标准建链请求后，严格遵循通信库定义的 RDMA 链路建立流程，完成与 RDMA 代理进程 2 的最终链路协商与认证，标志着整个分段式 RDMA 通信链路构建完成。

4.2.7 跨域 RoCEv2 通信自适应机制

为了解决广域网在丢包率、带宽保证、时延大小等能力的差异，设计了一种适应复杂广域网的跨域 RoCEv2 流量传输方式和带宽的自适应机制，以提高 RoCEv2 传输效率。

跨域通信的集群都需要部署 RoCEv2 网关，RoCEv2 网关负责 RoCEv2 跨域通信；控制中心负责根据网关能力和广域网能力协调 RoCEv2 跨域通信的传输方式和带宽需求。其工作流程如图 21 所示。

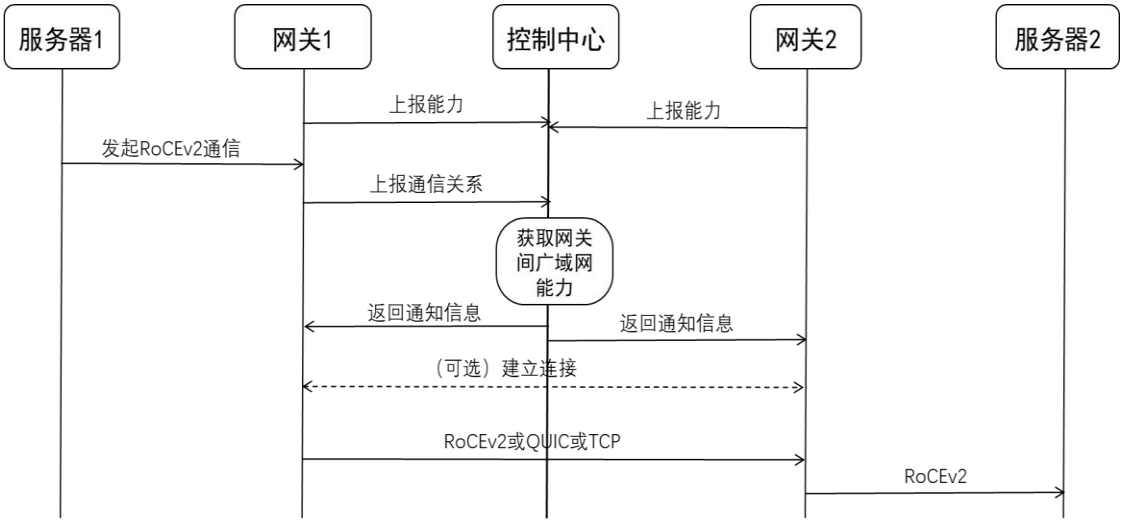

图 21　跨域 RoCEv2 流量传输方式和带宽的自适应工作流程

网关完成部署与启动流程后，需向控制中心进行能力参数注册，具体涵盖：

- RoCEv2 协议代答及 CNP 机制支持状态，以及对应链路带宽吞吐性能指标；
- QUIC 代理功能支持状态及其带宽处理能力；
- TCP 代理功能支持状态及其带宽处理能力。

控制中心在完成网关能力参数建档后，同步整合广域网传输性能参数，包括：

- 互联网链路的端到端时延特性；
- 确定性网络的带宽容量、时延指标、抖动幅度及丢包率统计；
- 带宽预留网络的带宽配置、时延表现、抖动情况及丢包率参数。

在服务器跨域通信场景中，源服务器（服务器 1）发送 RoCEv2 协议数据流。源网关（网关 1）接收到数据帧后，通过解析数据包头部获取源 IP 地址、源端口号、目的 IP 地址及目的端口号等四元组信息，并将通信会话元数据上报至控制中心。控制中心基于接收到的通信元数据，在网关能力数据库中检索源网关（网关 1）与目的网关（网关 2）的适配通信能力参数，同时结合广域网性能参数库，通过以下渠道获取链路状态信息：

- 确定性广域网控制器周期性上报的链路状态数据；
- 广域网控制器采集的网络性能参数；
- 网关间主动探测获取的实时链路状态信息。

控制中心基于上述信息，执行通信路径决策与带宽资源规划：在确定网关间最优通信协议后，计算并下发带宽配置参数。具体计算模型如下：

- 设原始业务带宽需求为 n，网关间链路有效传输率为 a，广域网链路有效传输率为 b；
- 则网关侧带宽配置需求为：a/n；
- 广域网边缘设备（PE）带宽配置需求为：n/(a×b)。

源网关与目的网关根据控制中心下发的通信协议指令及带宽配置参数，完成资源预分配流程。若采用 TCP 或 QUIC 代理协议，需建立代理连接通道，对 RoCEv2 数据进行协议转换与缓存处理，并通过代理通道实现数据转发。

4.2.8 拓扑感知的 Rank 号规划

在大模型跨域训练构建的复杂算网拓扑环境下，传统的静态 RANK 分配机制已显露出显著的不适应性。该机制基于预设的固定计算节点序列分配 RANK 号，在训练过程中无法动态调整。然而，跨域训练涉及多地域、多集群的异构计算资源，网络延迟波动、节点负载不均、动态资源调度等问题频发。当某个计算节点出现性能瓶颈或网络故障时，静态 RANK 分配无法灵活重组计算链路，极易导致训练效率大幅下降，甚至引发训练中断，难以满足大模型训练对资源动态调度和高容错性的要求。为此，设计了一种面向大模型跨广域训练的 RANK 号分配与管理方法，以满足大模型通过广域网跨集群训

练的场景。

增加一个全局的 RANK 管理模块来实现 RANK 的自动化管理，如图 22 所示。RANK 管理模块从算力调度器接收模型切分结果（包括租户、任务号、每个集群的 RANK 范围，RANK 间通信质量要求），然后根据模型切分结果为所有容器分配 RANK 号、选取 MASTER 并将结果（RANK 号和 MASTER 的 IP 地址）通知所有 RANK。之后将 RANK 间通信质量要求转化为 IP 间的通信质量要求通知给网络调度器。

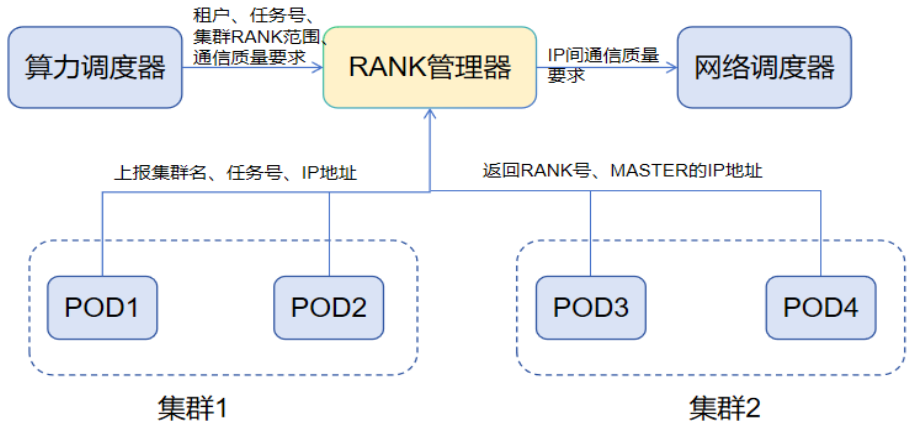

图 22　拓扑感知的 Rank 规划整体架构

第一个 RANK 号段（0 到 N）的集群中，第一个上报信息的 POD 分配 RANK 号 0，作为 MASTER；第二个上报信息的 POD 分配为 1，以此类推。其他 RANK 号段（N+1 到 N+M）的集群中，第一个上报信息的 POD 分配号为 N+1，　第二个上报信息的 POD 分配为 N+2，以此类推。

当每个 POD 有多个网卡（多个 IP 时），根据网卡名排序后选择第一个 IP 分配最小的 RANK 号；第一个 RANK 号段中最后一个

RANK 和第二个号段中第一个 RANK 之间存在跨域通信关系，根据网卡名排序后确定一对一的通信关系。

对于一个需要三个 POD 的大模型训练任务，算力调度器根据计算结果将其调度到两个集群：集群 1 和集群 2。其中，集群 1 调度了两个 POD（POD1 和 POD2）；集群 2 调度了一个 POD（POD3）如图 23 所示。

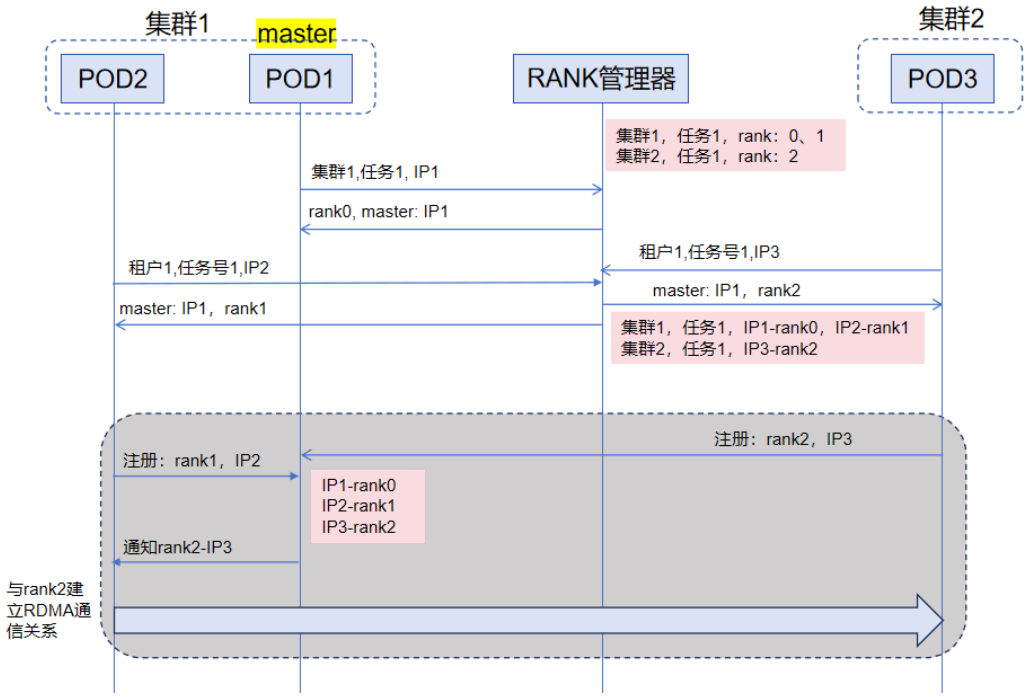

图 23　RANK 管理流程图

4.3　异属合训

4.3.1　多队列排队协作技术

当训练作业的子任务分散在分属不同运营主体的算力中心时，由于各异属队列的调度规则、资源分配策略相互独立，难以实现子作业

状态的同步与协同，导致大模型作业无法同步获得分配算力资源。为解决多主体、多队列场景下大模型训练作业协同调度难题，提出多异属队列协作机制，支持将大模型训练作业拆分后放入不同主体队列同步排队，确保子作业同步获资源，减少排队时间，避免资源死锁，打破算力壁垒，提升跨域训练效率。

大模型跨域训练任务拆分为子任务集，分发至多个集群 / 平台同步启动。池化调度依据任务需求与资源状态拆分模型，选定方案后分发部署需求，各平台将其加入作业队列并按策略调度。

因各集群平台队列独立调度，同一训练业务任务调度时机不一，易造成资源浪费、效率降低甚至资源互锁。为此，构建跨域多任务队列协调器，协调多队列作业调度，实现跨域作业同步运行。

跨域多队列协调器作为负载控制器的一部分，负责获取作业分散于各平台队列的状态，确保同一训练作业任务同步获资源。其目标是调整各子任务位置，避免资源互锁，分布于全局管控层与异属资源层，包含异属队列协调器、任务代理、异属队列控制器：

- 异属队列协调器。基于各异属队列状态（包括各异属队列可用资源量、各异属队列中作业的子部分排队次序与资源需求），对全局作业队列的排队次序进行调整。
- 任务代理。从异属队列控制器获取排队信息，形成当前所在异属队列中作业的排队次序，与所有异属队列的资源可用量、各任务子部分资源需求量一并上报。另外，监听全局队列的状态，若不一致时，通过调整本地队列排队次序与全局队列

次序一致。

- 异属队列控制器。为任务代理提供本队列的状态信息。同时，接收来自任务代理的排队状态调整请求，配合完成本队列的任务排队次序调整。

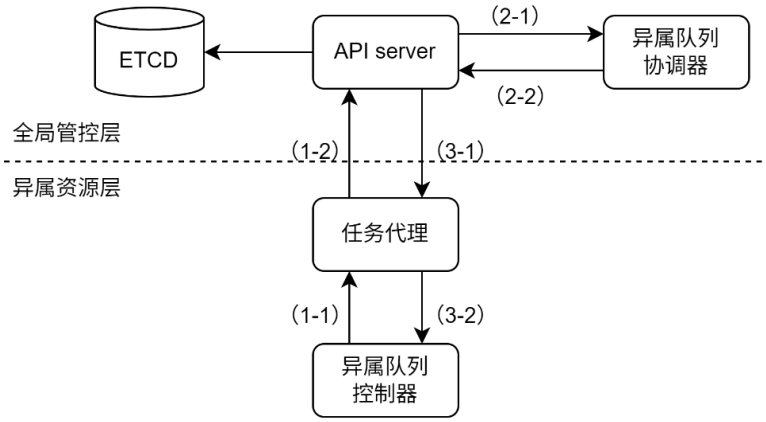

图 24　多队列排队协作架构

为实现跨多队列协调器调整各异属队列的能力，需各异属队列提供其队列内作业的排队状态以及调整其队列内作业排队顺序的接口与权限。在这些前提条件具备后，跨多队列协调器的工作机制如下：

- 周期上报作业排队状态。任务代理器实时获取各自所在异属平台上作业排队状态，上报给任务调度。
- 更新预期排队状态。任务调度中异属作业队列协调器，根据作业排队现状判断是否调整预期排队状态。
- 按预期排队状态调整队列。任务代理器监听预期作业排队状态的变化，根据预期作业排队对所在异属平台的作业队列进行调整。

管控层可获取各个异属平台中任务代理上报各自的作业排队次

序、各作业的资源需求量与各异属队列的可用资源量。除此之外，管控层会维护一个全局队列，资源层的各异属平台中的队列次序需参照全局队列次序进行相应的调整。如图 25 所示，共有 4 个作业分别为 Job1~Job4。有 3 个异属队列分别为 queue1~queue3。在管控层会维护一个全局队列 Global Queue。初始时，作业可根据作业提交的先后顺序入队，经管控层的调度，各作业的分别情况为：

- Job1 分配到 queue1。
- Job2 拆分为 Part1 和 Part 2，分配到 queue1 和 queue2。
- Job3 拆分为 Part1 和 Part2，分配到 queue2 和 queue3。
- Job4 分配到 queue4。

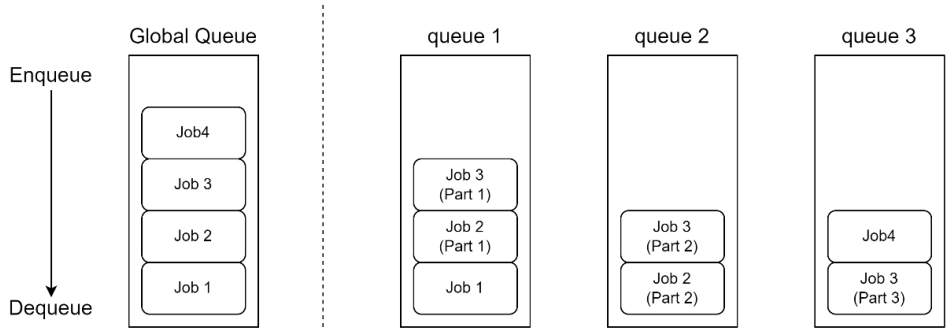

图 25　多队列排队示意图

作业排队状态的调整，本质上是依据各异属队列中作业各子部分的排队次序、子部分的资源需求量，以及各个异属队列的可用资源量，按照特定策略目标配置全局队列的作业排队次序，从而确定预期的作业排队状态。各异属平台的任务代理会实时监听全局队列状态，一旦全局队列状态发生变化，任务代理便会将所在异属队列的排队次序与全局队列进行比对。若两者不一致，任务代理将通过调整作业信息、

队列信息、调度策略等指标项，确保所在异属队列的排队次序与全局队列保持同步。

如各异属队列的资源可用量与作业的需求量如下：

- 异属队列 queue1、queue2、queue3 当前资源可用量均为 6；
- 作业 Job1 资源需求量为 8；
- 作业 Job2 资源需求量分为两部分，Part1 为 5、Part2 为 5；
- 作业 Job3 资源需求量分为三部分，每部分均为 6；
- 作业 Job4 资源需求量为 4。

在以提升综合资源利用率为目标时，Job1、Job2、Job3、Job4 的排队次序应调整为 Job3、Job2、Job4、Job1。通过这种排序，能够实现各个异属队列综合资源利用率的最大化。

若以提升综合作业吞吐量为目标，这四个作业的次序则需调整为 Job4、Job2、Job3、Job1。此排列方式可使各异属队列的综合作业吞吐量达到最优。

当以保障作业 deadline 为目标时，若存在有严格 deadline 的作业，需优先保障其资源分配。例如，若 Job1 存在严格 deadline，作业次序应调整为 Job1、Job2、Job4、Job3。这样既能确保 Job1 优先获得资源，Job4 也可获取部分资源，而 Job2 和 Job3 则因 Job1 占用资源而继续处于排队状态 。

4.3.2 多队列联合抢占技术

由于各智算中心通常通过独立的管控平台对外提供算力服务，导

致本地管控平台需要同时处理两类作业请求：跨集群协作作业与本地用户提交的作业。当资源供给无法满足需求时，抢占机制成为保障高优先级作业执行的关键手段。然而，传统的单集群抢占策略存在显著局限性，其缺乏跨中心的有效协调机制，极易造成子作业启动时序不一致，严重影响分布式训练的整体效率。

为破解跨多智算中心场景下作业抢占的协同难题，针对性地提出一套多异属队列联合抢占的技术架构。在此基础上，设计了大模型训练作业跨多异属队列联合抢占与被抢占同步方法。跨广域多集群异属队列抢占调度的技术架构如图 26 所示。

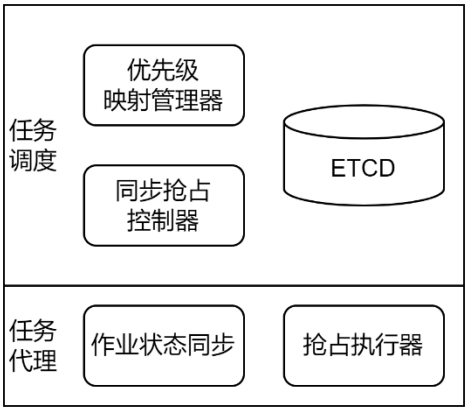

图 26　多队列联合抢占功能模块图

跨域调度系统由任务调度与任务代理构成。任务调度负责维护跨集群资源视图，存储各智算中心的实时资源状态，实现作业拆分策略，生成子作业与资源需求描述，执行全局抢占决策，下发抢占指令至本地调度器。任务代理对接各智算中心管控平台，接收全局调度器下发的抢占策略，在本地资源分配时执行抢占动作，并反馈抢占执行状态。为实现跨域多异属队列的抢占机制，需实现分别在任务调度与任务代

理分别实现如下功能模块。

- **优先级映射管理器**

优先级定义了任务的紧急程度。高优先级任务可以抢占低优先级任务的资源。异属队列的优先级设计各种独立，为实现异属队列的抢占协作，需建立统一的优先级规范。该功能模块的功能就是维护全局的优先级映射表，将所纳管的异属队列的优先级与全局优先级形成映射，建立统一的优先级规则。

- **同步抢占控制器**

子作业任务调度会将大模型训练作业拆分为多个子作业部署在合适的智算中心。各智算中心会各自的调度策略进行子作业的调度，将导致同属作业的子作业状态不一。分布在多个智算中心，各智算中心的抢占时机不一致。对于同一作业而言，其所有子作业均获得运行所需资源才能正常训练。为此，需在任务调度协调各异属队列的抢占时机，实现作业同步抢占、同步运行。

- **作业状态同步**

该部分功能在任务代理上实现，将本地的作业状态上报到任务调度，实现全局与本地智算中心的作业状态同步。

- **抢占执行器**

同步抢占控制器作为全局的抢占协调器，而抢占执行器则作为任务代理器上实际异属队列抢占的触发器。在所有子作业的抢占状态就绪后，同步抢占控制器将作业抢占状态置为可抢占状态。抢占执行器将可抢占状态的子作业通知异属队列的调度器，由异属队列调度器执

行实际的抢占。

（1） 优先级映射管理

优先级映射管理的主要功能是维护全局优先级与各异属队列优先级的映射表，如表 2 所示。

表 2 多队列优先级映射关系

全局 优先级	异属队列 1 优先级	异属队列 2 优先级	异属队列 3 优先级
	Super/Root		P1
High	Administrator	VIP 3	P2
Middle	Manager	VIP 2	P3
Low	NormalUser	VIP 1	P4
	AnonymousUser	VIP 0	P5

该优先级映射表在异属智算中心注册纳管时需将信息录入，后续在将调度后产生的子作业进行渲染时，需参考此表将优先级转化为对应智算中心的优先级描述。全局优先级与各智算中心的优先级映射需同时考虑租户间的优先级与作业间的优先级，以保障全局租户/作业优先级的一致性。

（2） 多队列同步抢占

此部分需任务调度与任务代理协作完成。在任务调度将子作业分发到各智算中心后，各智算中心的任务代理将承接这些子作业，管理子作业的生命周期，与全局的任务调度协作使子作业在本地智算中心顺利完成。在异属队列抢占过程中，需实现的功能由两部分构成：全

局/本地子作业状态同步、异属多队列的抢占时机同步，具体的功能模块间交互关系如图 27 所示。

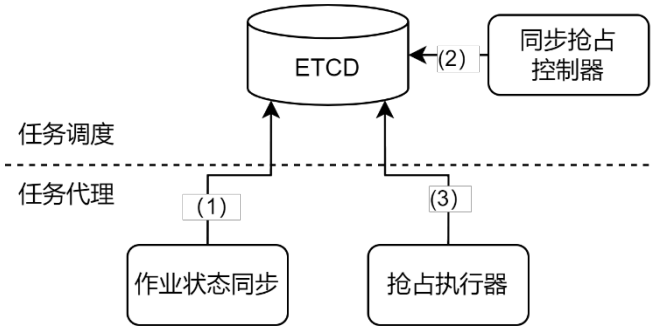

图 27　多队列抢占功能模块

- 将各子作业状态同步到全局任务调度。如预抢占，即抢占就绪状态，表示该子作业所在的智算中心有可被该子作业抢占的低优先级任务。被抢占状态则表明该子作业已被抢占，将其状态同步给全局的任务调度，以便全局任务调度将此状态同步到其他同属一作业的子作业所在队列。
- 获取同属一作业的各子作业抢占就绪状态。若所有子作业均就绪，则表明该作业可执行抢占，更新子作业状态为可抢占。
- 当子作业为可抢占时，表明该子作业可抢占本地的低优先级作业。抢占执行器获取到执行抢占的子作业后，触发本地的队列调度器执行抢占。当子作业为被抢占状态时，抢占执行器将通知本地的调度器驱逐该子作业并保护抢占现场。

（3） 跨域抢占保护

跨域抢占保护旨在保证抢占执行全过程中业务的安全行与完整性，降低抢占带来的开销。此功能分为抢占的现场保护与现场恢复。

- 抢占现场保护

在确定可执行抢占时，需先由调度器确定需要被驱逐的任务。智算中心本地将对这些任务状态进行快照。如大模型训练任务的CheckPoint、运行时相关参数、数据处理进度等。与目前本地现场保护不同的是相关现在保护的状态数据需具备全局数据视图与跨域状态数据同步能力。与此同时，需将被抢占状态同步给全局的任务调度，进而通知部署了其他同属同一作业的子作业智算中心进行跨域的资源抢占。

- 抢占现场恢复。

现场恢复的过程与现场保护相反。被抢占的任务会被本地调度加入到待调度队列，获得重新调度的机会。当该任务重新获得资源时，将从现场保护阶段保存的快照状态恢复训练。若后续任务调度部分实现了跨域重新调度逻辑，该任务可以通过全局的任务调度获得重新调度机会，在其他智算中心部署运行。

（4） 异属队列抢占与被抢占流程

- 抢占流程

跨域训练场景下，大模型训练作业经调度后会分布在多个智算集群，在发起抢占时，需分布在多个智算中心的各个子作业需都满足抢占条件。当某个智算中心的子作业满足抢占条件时，需先将预抢占状态通知给全局的任务调度，待所有子任务均具备抢占条件时再通知各智算中心的异属队列执行抢占。

- 被抢占流程

分布在各智算中心的子作业能够抢占低优先级的作业，同样可被更高优先级的作业抢占。当某个智算中心的子作业被抢占时，需将被抢占状态上报全局的任务调度。当任务调度获知任意一个子作业被抢占时，需通知与该子作业同属一作业的其他子作业释放资源。

4.3.3 RDMA 网络虚拟化

为了解决异属数据中心底层网络独立规划所导致的 IP 地址互相冲突的情况，满足大模型跨域协同训练的通信需求，提出异属 RDMA over VxLAN 技术方案，旨在构建兼顾严格隔离与高效传输的跨域通信机制，聚焦于安全隔离、性能优化与场景适配三个维度：

- 安全隔离：构建由虚拟网络标识（VNI）、虚拟局域网（VLAN）、安全访问控制组（ACL），确保跨数据中心 RDMA 流量在多租户环境下实现端到端的逻辑隔离与数据安全传输。
- 性能优化：采用 RDMA 感知调度算法结合轻量化 VxLAN 封装协议，在保障网络低延迟特性的前提下，将 RDMA 数据传输吞吐量提升 30%以上，实现与高性能计算场景的深度适配。
- 场景适配：设计支持传统广域网、确定性网络等异构网络环境的弹性接入机制，通过标准化接口协议实现与不同主体数据中心网络架构的无缝兼容，提升方案跨域部署能力。

（1） VxLAN 异属强化

在原有 VNI-VLAN 映射基础上，进一步强化跨主体场景下的隔离与扩展能力。除 VNI-VLAN 映射外，为每个租户配置独立的安全

组规则，限制其 RDMA 流量的源/目的端口范围。新增租户时，系统自动完成“VNI 分配→VLAN 子接口创建→RDMA 参数配置”的全流程自动化：控制器根据租户的 RDMA 需求（如带宽、并发连接数），预配置网关的队列资源与转发策略，整个过程无需调整物理网络拓扑，支持分钟级租户上线。

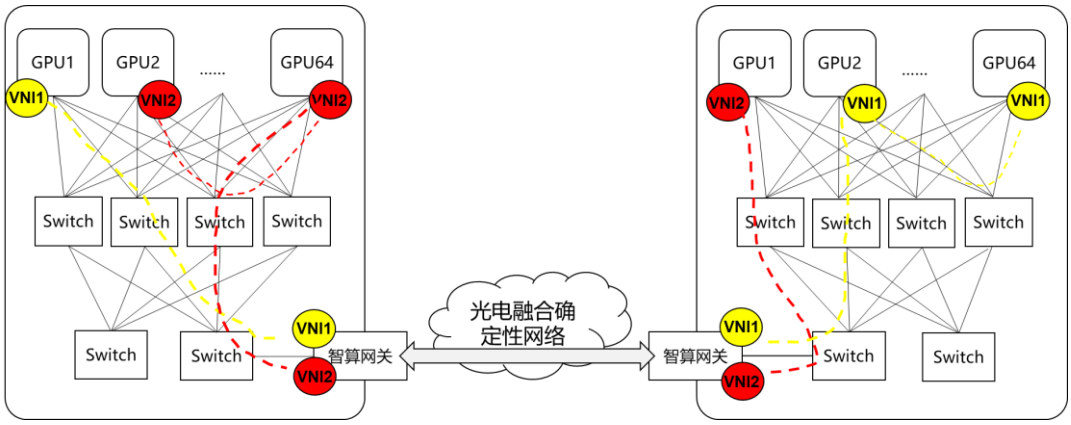

图 28　异属 VxLAN 互联拓扑结构

如图 28 所示，智算网关作为连接 VxLAN 虚拟网络与传统物理网络的关键节点，在原有 VTEP（VxLAN 隧道端点）功能基础上，新增 RDMA 适配模块，形成“封装转换+协议优化+隔离管控”的三位一体架构。

接收来自租户计算节点 VTEP 的 VxLAN 封装流量时，通过深度包检测（DPI）识别 RDMA 协议特征（如 RoCEv2 的 UDP 端口 4791），为这类高优先级流量开辟专用处理通道，绕过传统 TCP/IP 协议栈的冗余校验环节，降低封装/解封装延迟。同时，保留 VxLA 头部的 VNI 标识，确保租户隔离属性不丢失。

在 VNI-VLAN 一一映射的基础上，为每个映射关系绑定 RDMA

流量的服务质量参数。例如，为高优先级租户（如大模型训练任务）配置 “低延迟队列+带宽预留”策略，当解封装后的 RDMA 流量通过 VLAN 子接口转发时，自动触发队列调度机制，保障跨域传输的确定性。此外，支持动态映射调整，当某租户临时需要扩容跨域带宽时，可通过控制器远程更新 VNI 对应的 VLAN 子接口带宽阈值。

网关内置 RDMA 会话跟踪模块，记录租户 RDMA 连接的源/目的地址、QP（队列对）状态等信息。当跨数据中心的 RDMA 流量经过时，通过会话信息验证确保流量合法性，避免未授权的跨主体 RDMA 访问。同时，针对 RDMA 的拥塞通知（如 CNP 报文）进行特殊处理，在 VxLAN 封装与解封装过程中保留拥塞标记，确保跨域场景下 RDMA 的动态速率调整机制正常生效。

（2） 轻量级 RDMA 封装适配

为解决 VxLAN 封装对 RDMA 性能的损耗，方案从协议适配与路径优化两方面进行针对性设计。

针对 RDMA 零拷贝特性，网关的 VxLAN 封装与解封装过程采用 “内存直透”技术：接收计算节点 RDMA 内存中的数据时，直接在用户态完成 VxLAN 头部的添加/剥离。该机制避免了数据在内核态与用户态之间的拷贝，显著降低了单包处理延迟，满足 RDMA 对低延迟的要求。在报文封装过程中，通过特定的内存映射与操作接口，直接将 RDMA 数据与 VxLAN 头部进行整合，形成完整的 VxLAN 报文；解封装时则反向操作，精准剥离 VxLAN 头部，将原始 RDMA 数据快速交付上层应用，减少不必要的处理环节。

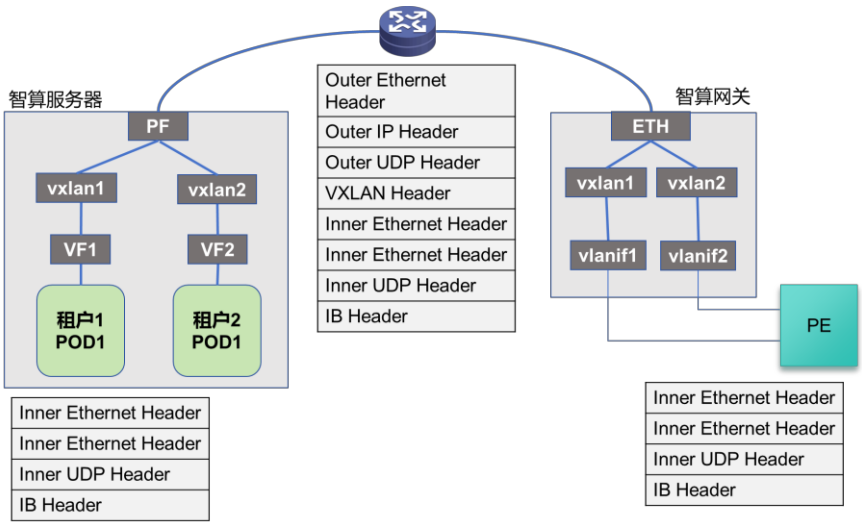

图 29　RDMA over VxLAN 报文封装

RDMA 网关通过网络调度与广域网控制器联动，实时获取跨数据中心链路的带宽、时延、丢包率等参数。当检测到链路拥塞时，自动触发 RDMA 流量的动态调整：对于 RoCEv2 流量，通过修改其 DSCP 标记优先占用低延迟链路；对于需要重传的数据包，临时切换至 TCP 代理模式，避免 RDMA 原生重传机制在高丢包场景下的性能劣化。在报文处理上，对于动态调整后的流量，根据不同传输模式重新封装报文，如切换到 TCP 代理模式时，将 RDMA 数据适配到 TCP 协议的报文格式中进行传输，确保数据在复杂网络环境下高效、稳定传输 。

（3） 高效 VxLAN 卸载

以 Open vSwitch（OVS）的 datapath 作为慢路径，通过深度整合 RDMA 网卡的增强型虚拟交换机，构建软硬协同的流量处理机制。该机制基于 eSwitch 的硬件加速能力，实现 VxLAN 报文的快速解析与转发，同时将复杂的协议处理流程卸载至网卡硬件，有效降低

CPU 负载，显著提升网络流量转发效率。

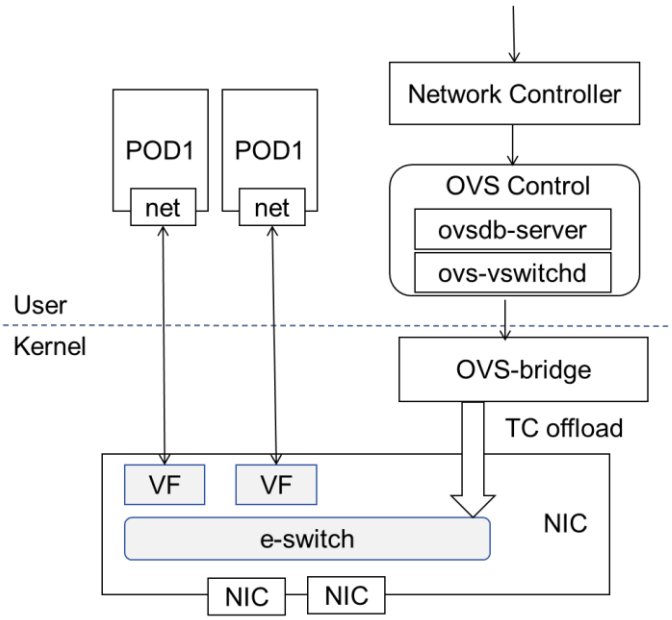

图 30　VxLAN 硬件卸载技术架构

初始时向 eSwitch 预置一条缺省匹配规则。当首包抵达 eSwitch 时，由于尚未建立与之匹配的流表项，将触发该缺省规则。在此机制下，报文通过 vf-representor 导向 eSwitch 的管理端口，并进一步传递至对应 OVS 的数据路径。由于 datapath 内同样缺乏匹配表项，报文将借助内核的 netlink 通信机制，上传至 OVS 用户态进程 ovs-vswitchd 进行后续处理。

ovs-vswitchd 作为控制平面的核心组件，存储着由 OpenFlow 协议下发的流转发规则，能够实现首包的精准转发。与此同时，该进程将对当前数据流对应的规则进行深度分析，依据预设条件判断其是否满足卸载至 RDMA 网卡的技术要求。若判定规则符合卸载标准，ovs-vswitchd 将通过 TC 接口，将该流规则推送至 eSwitch 的硬件转发单元。

对于同一数据流的后续报文，当其到达 eSwitch 时，将直接匹配已部署的硬件流表项。这种设计使得报文无需经过主机操作系统的内核态与用户态处理流程，即可在 RDMA 网卡的 eSwitch 中完成快速转发。此过程有效规避了传统软件转发的性能瓶颈，充分释放了 RDMA 网卡的高带宽与低时延特性，显著提升了 VxLAN 网络环境下的流量转发效率，尤其适用于大规模分布式训练等对网络性能要求极高的应用场景。

为保障规则卸载机制的可靠性与兼容性，本方案引入动态规则更新策略。当控制平面的 OpenFlow 规则发生变更时，ovs-vswitchd 将通过实时监测机制感知变化，并同步更新 eSwitch 硬件中的流表项，确保报文转发策略的一致性与时效性。此外，针对包含复杂访问控制逻辑或协议转换需求的特殊规则，系统将自动启用回退机制，通过 OVS 的数据路径与 ovs-vswitchd 协同处理，实现硬件加速与软件处理相结合的混合转发模式。

五、 验证与评估

5.1 试验环境

试验环境由异属异构异地三个算力集群和一个总控集群组成，如图 31 所示。每个算力集群包含若干台算力服务器，如 H20 GPU 服务器、天垓 150（BI150）GPU 服务器。这些算力服务器均由本集群的

算力资源管理系统统一管理。P 设备通过 2 条 2000 公里光纤分别与 PE-2、PE-3 设备连接。P 设备与其他设备之间均通过短距光纤连接。在这种网络拓扑下，任意两个算力集群之间均可通过大于 2000 公里的广域网链路进行协同训练。

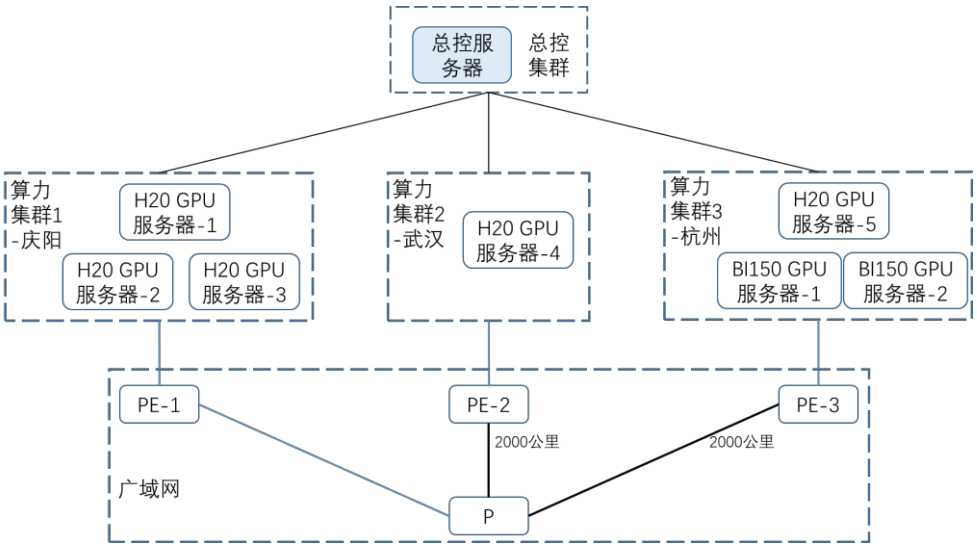

图 31　试验环境拓扑

5.2　测试验证

5.2.1　异属算力集群协同训练能力验证

（1）向调度系统提交训练任务 1，如表 3 所示，等待调度系统的反馈。选择在集群 1 运行训练任务 1，最后观察训练任务 1 的日志，如图 32 所示，根据日志计算出训练任务 1 的训练性能数据，如表 4 所示。

表 3 训练任务 1 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B

GPU 数量	24 卡 H20
并行设置	TP=4，PP=6，DP=1，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，不启用计算通信重叠
算网协同设置	不使用异属算力，不使用算网协同，不使用异构算力芯片

图 32　训练任务 1 的日志

表 4 训练任务 1 的训练性能数据

每迭代完成时间	60.7 秒
TGS（Tokens/gpu/s）	337.39
Samples/s	0.4942

（2）缩减集群 1（庆阳数据中心）中的可用 H20 GPU 算力资源。向调度系统提交训练任务 2（与训练任务 1 相同，如表 3 所示），等待调度系统的反馈。由于没有任何一个算力集群有 3 台可用的 8 卡 H20 服务器，调度系统此时提示跨域调度模式未激活，无法在同一个集群中完成训练，作业在排队中，如图 33 所示。

图 33　调度系统提示训练任务 2 需要排队

测试结果分析：如果用户不指定允许使用异属算力，当没有任何一个算力集群可以满足用户的全部算力资源需求时，训练任务无法运行。

（3）更改训练任务 2 描述文件，设置允许使用异属算力。向调度系统提交训练任务 3，如表 5 所示，等待调度系统的反馈。在调度系统反馈可用调度方案后，在集群 1（庆阳数据中心）上启动一个 POD 继续占用掉集群 1 的可用 GPU 资源，然后用户从可用调度方案中选择在集群 1（庆阳数据中心）和集群 3（杭州数据中心）运行训练任务 3。由于集群 1（庆阳数据中心）的可用 GPU 资源在用户选择可用调度方案前被其他用户占用掉，导致用户选择的可用调度方案失效，任务无法部署（如图 1 所示）。

表 5 训练任务 3、4、5、6 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B
GPU 数量	24 卡 H20
并行设置	TP=4，PP=6，DP=1，EP=1

其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，不启用计算通信重叠
算网协同设置	使用异属算力，不使用算网协同，不使用异构算力芯片

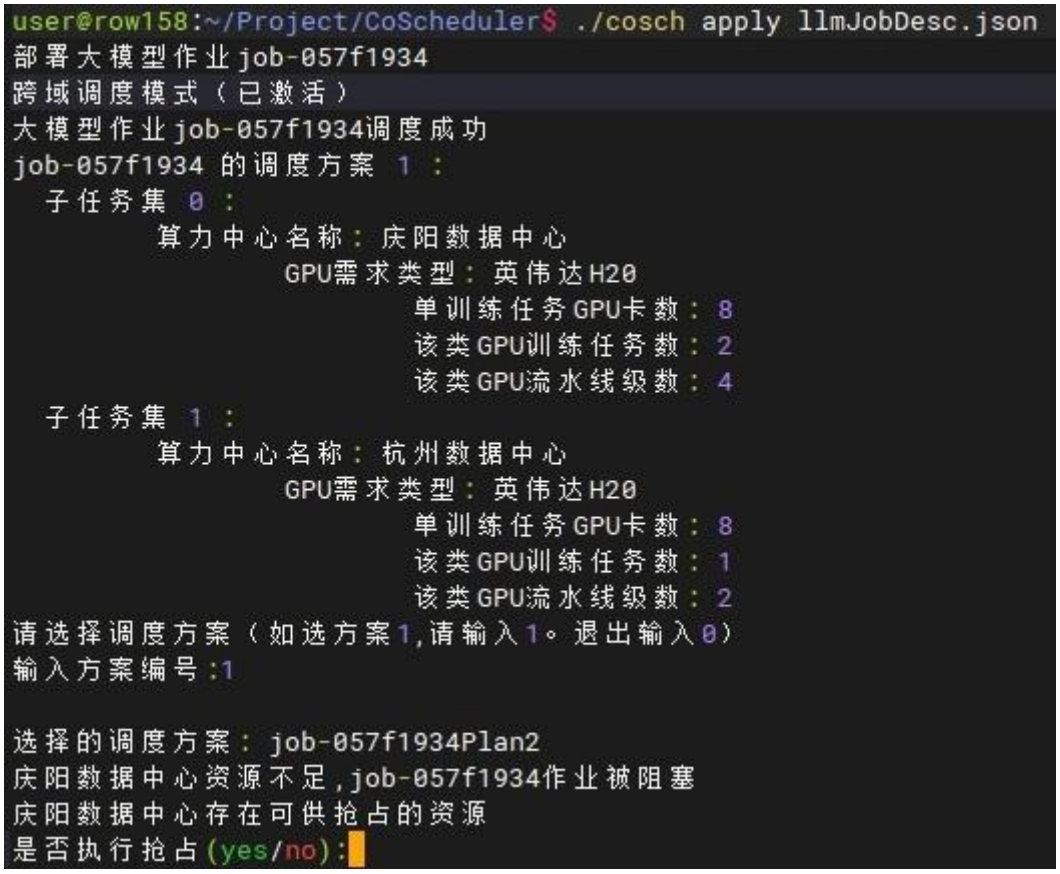

图 34　调度系统提示训练任务 3 部署失败

同时，调度系统提示用户集群 1 有可抢占的 GPU 资源，询问用户是否抢占。选择同意抢占后，调度系统发起抢占，在抢占成功后，训练任务 3 部署成功，如图 35 所示。最后观察训练任务 3 的日志，如图 36 所示，根据日志计算出训练性能数据，如表 6 所示。

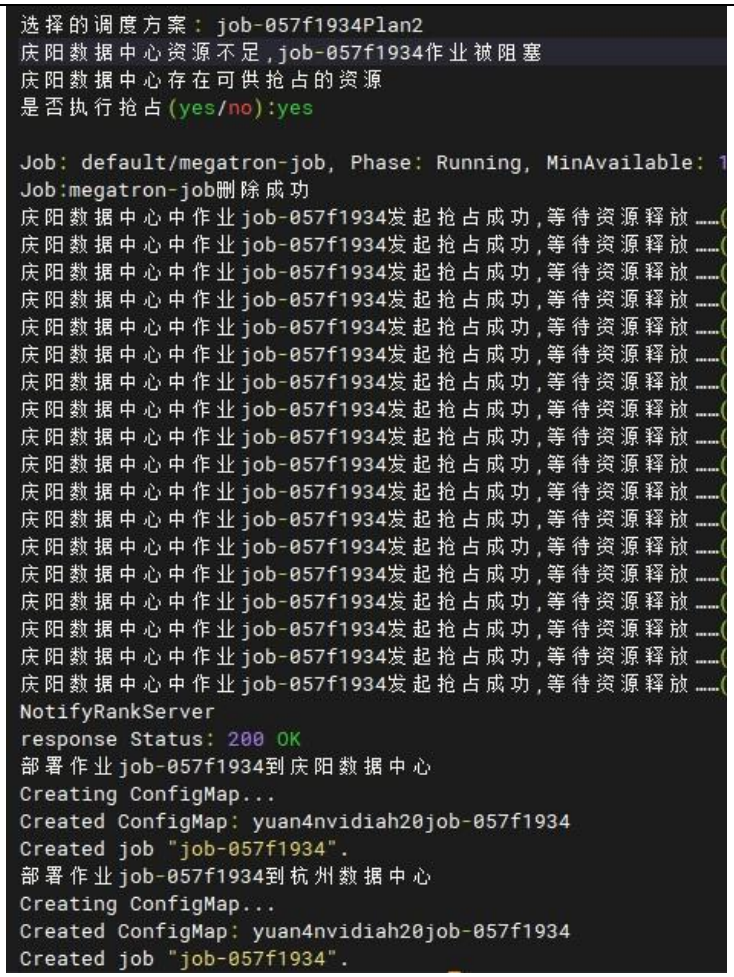

图 35　训练任务 3 抢占算力资源成功

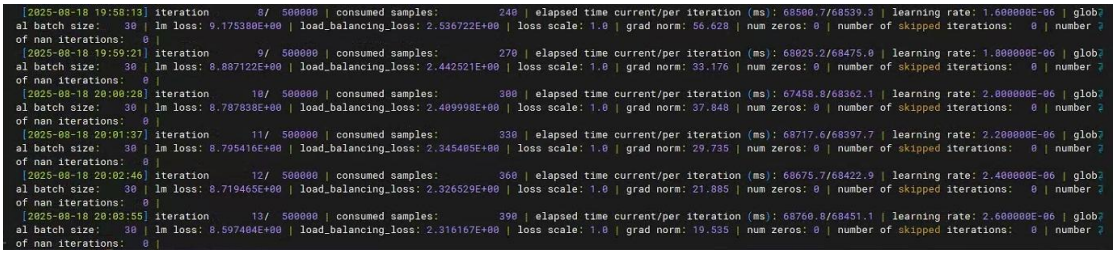

图 36　训练任务 3 的日志

表 6 训练任务 3 的训练性能数据

每迭代完成时间	68.4 秒
TGS（Tokens/gpu/s）	299.42
Samples/s	0.4386

测试结果分析：1、异属队列协作机制允许用户通过抢占尽快运

行训练任务；2、在 2000 公里以上广域网环境下，跨域训练性能下降。相对于训练任务 1，训练任 3 每迭代完成时间增加了 12.68%。训练任务 3 的 TGS 是训练任务 1 的 TGS 的 88.75%，即此时跨域训练效率是 88.75%。

5.2.2 广域确定性网络传输能力验证

（1）向调度系统提交训练任务 4，如表 5 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 4。在训练任务 4 运行过程中，在广域网链路上加入干扰流。最后观察训练任务 4 的日志，如图 37 所示，根据日志计算出训练任务 4 的训练性能数据，如表 7 所示。

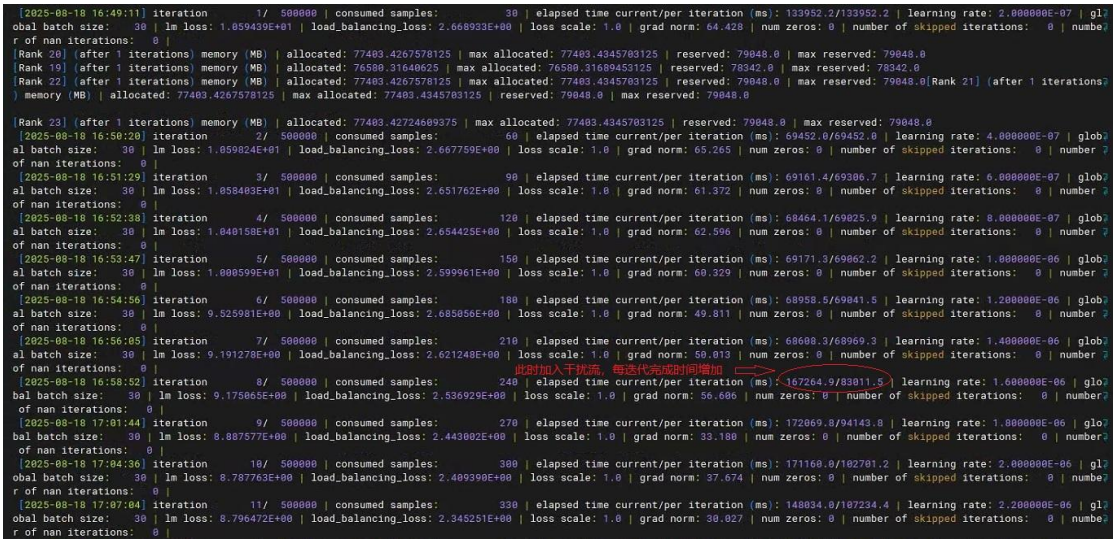

图 37　训练任务 4 的日志

表 7 训练任务 4 的训练性能数据

每迭代完成时间	171.1 秒
TGS（Tokens/gpu/s）	119.70

Samples/s	0.1753

测试结果分析：由于没有使用广域确定性，加入干扰流后，训练性能下降。相对于训练任务 3，训练任务 4 每迭代完成时间增加了 150.15%，TGS 下降了 60.02%。

（2）在广域网链路上加入确定性网络控制面。向调度系统提交训练任务 5，如表 5 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 5。在训练任务 5 运行过程中，在广域网链路上加入干扰流。最后观察训练任务 5 的日志，如图 38 所示，根据日志计算出训练任务 5 的训练性能数据，如表 8 所示。

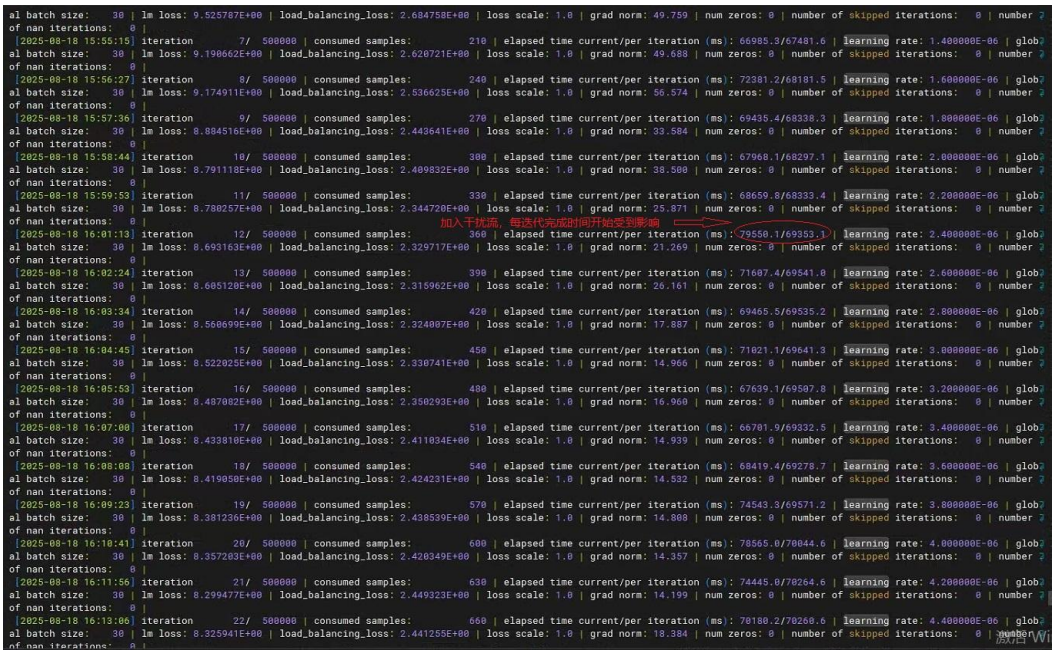

图 38 训练任务 5 的日志

表 8 训练任务 5 的训练性能数据

每迭代完成时间	70.2 秒
TGS（Tokens/gpu/s）	291.74

Samples/s	0.4274

测试结果分析：加入确定性网络控制面后，干扰流对训练性能的影响很小。相对于训练任务 3，训练任务 5 每迭代完成时间只增加了 2.63%，TGS 只下降了 2.56%。

5.2.3 异属算力与广域网络协同调度能力验证

（1）限制集群 1、2 之间的广域网链路带宽到 400Mbps。向调度系统提交训练任务 6，如表 5 所示，等待调度系统的反馈。调度系统反馈的可用调度方案中包含广域链路带宽很低的算力集群“1+2”组合，如图 39 训练任务 6 的可用调度方案 所示。选择在集群 1 和集群 2 运行训练任务 6。最后观察训练任务 6 的日志，如图 40 所示，根据日志计算出训练任务 6 的训练性能数据，如表 9 所示。

图 39 训练任务 6 的可用调度方案

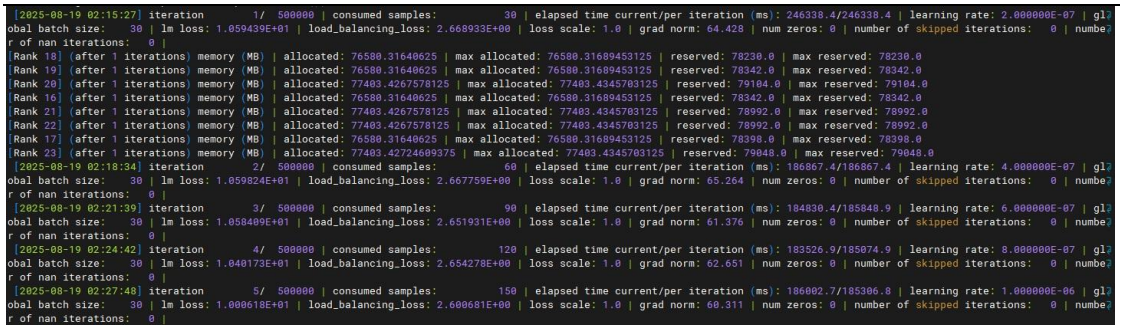

图 40 训练任务 6 的日志

表 9 训练任务 6 的训练性能数据

每迭代完成时间	185.3 秒
TGS（Tokens/gpu/s）	110.52 秒
Samples/s	0.1619

测试结果分析：由于没有开启算力协同调度，用户收到的可用调度方案中有可能会包含广域链路带宽很低的算力集群组合。此时用户恰好选择了这个广域链路带宽很低的算力集群组合，无法高效完成训练。相对于训练任务 3，训练任务 6 每迭代完成时间增加了 170.91%，TGS 下降了 63.09%。

（2）开启算网协同。向调度系统提交训练任务 7，如表 10 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 7，如图 41 所示。最后观察训练任务 7 的日志，如图 42 所示，根据日志计算出训练任务 7 的训练性能数据（如表 11 所示）。

表 10 训练任务 7 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B
GPU 数量	24 卡 H20

并行设置	TP=4，PP=6，DP=1，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，不启用计算通信重叠
算网协同设置	使用异属算力，使用算网协同，不使用异构算力芯片

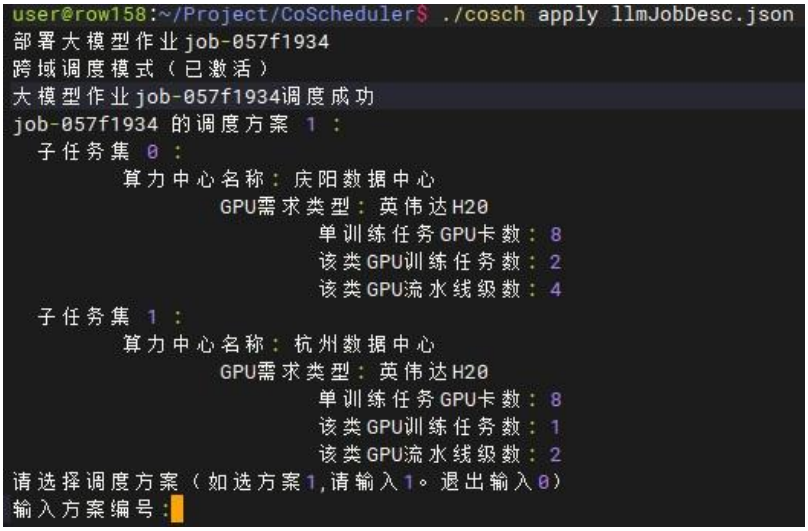

图 41 训练任务 7 的可用调度方案

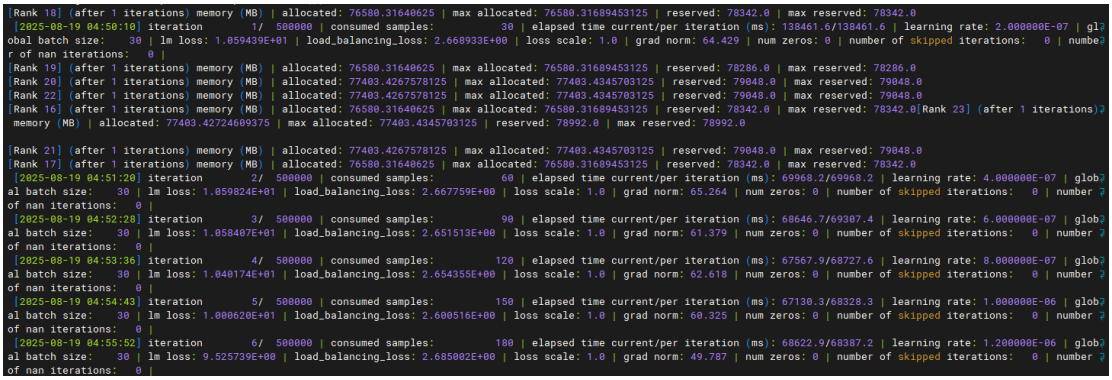

图 42 训练任务 7 的日志

表 11 训练任务 7 的训练性能数据

每迭代完成时间	68.4 秒
TGS（Tokens/gpu/s）	299.42
Samples/s	0.4386

测试结果分析：开启算力协同调度后，用户收到的可用调度方案中不会包含广域链路带宽很低的算力集群组合。此时跨域训练效率恢复正常。

5.2.4 计算与通信重叠的流水线并行训练能力验证

（1）开启计算通信重叠。向调度系统提交训练任务 8，如表 12 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 8，最后观察训练任务 8 的日志，如图 43 所示，根据日志计算出训练任务 8 的训练性能数据（如表 13 所示）。

表 12 训练任务 8 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B
GPU 数量	24 卡 H20
并行设置	TP=4，PP=6，DP=1，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，启用计算通信重叠
算网协同设置	使用异属算力，使用算网协同，不使用异构算力芯片

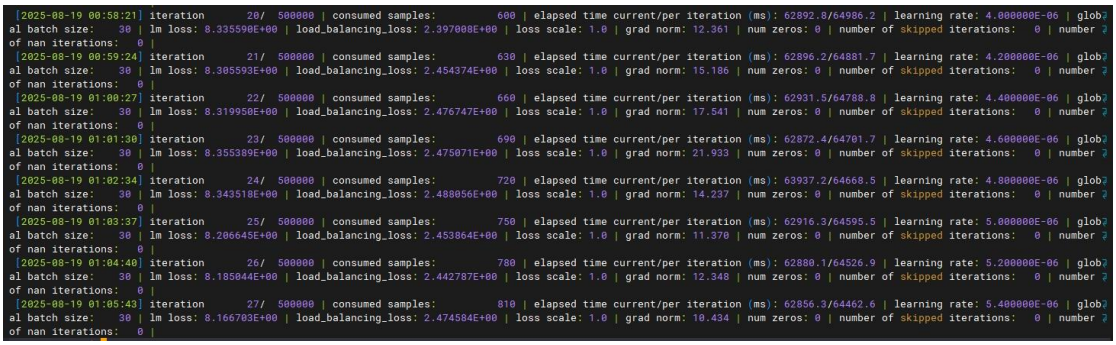

图 43 训练任务 8 的日志

表 13 训练任务 8 的训练性能数据

每迭代完成时间	64.4 秒
TGS（Tokens/gpu/s）	318.01
Samples/s	0.4658

测试结果分析：启用计算与重叠流水线后，跨域训练性能有提升。相对于训练任务 1，每迭代完成时间只增加了 6.10%。训练任务 8 的 TGS 是训练任务 1 的 TGS 的 94.26%，即此时跨域训练效率是 94.26%。

（2）增加全局批次大小。向调度系统提交训练任务 9，如表 14 所示，等待调度系统的反馈。选择在集群 1 运行训练任务 9，最后观察训练任务 9 的日志（如图 44），根据日志计算出训练任务 9 的训练性能数据（如表 15 所示）。

表 14 训练任务 9、10 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B
GPU 数量	24 卡 H20
并行设置	TP=4，PP=6，DP=1，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=128，浮点数精度 fp16，不启用计算通信重叠
算网协同设置	使用异属算力，使用算网协同，不使用异构算力芯片

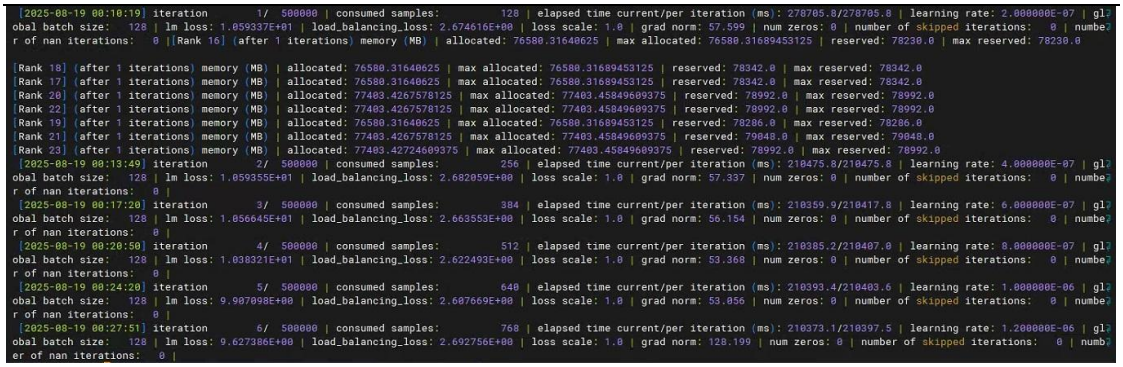

图 44　训练任务 9 的日志

表 15　训练任务 9 的训练性能数据

每迭代完成时间	210.4 秒
TGS（Tokens/gpu/s）	415.31
Samples/s	0.6084

（3）向调度系统提交训练任务 10，如表 14 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 10，最后观察训练任务 10 的日志，如图 14 所示，根据日志计算出训练任务 10 的训练性能数据，如表 16 所示。

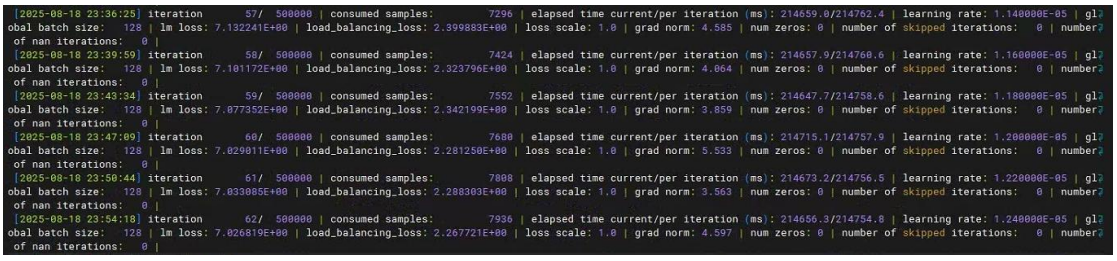

图 45　训练任务 10 的日志

表 16　训练任务 14 的训练性能数据

每迭代完成时间	214.8 秒
TGS（Tokens/gpu/s）	406.80

Samples/s	0.5959

测试结果分析：由于全局批次大小增加，训练任务 10 的 TGS 是训练任务 9 的 TGS 的 97.95%，即此时跨域效率约为 98%。

5.2.5 异构算力芯片混合训练能力验证

设置允许异构算力芯片。向调度系统提交训练任务 11，如表 17 所示，等待协同调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 11，最后观察训练任务 11 的日志，如图 46 训练任务 11 的日志所示，根据日志计算出训练任务 11 的训练性能数据，如表 18 所示。

表 17 训练任务 11 的描述信息

大模型	Mixtral 8x7B，总层数 70 层，总参数量 101B
GPU 数量	24 卡 H20+4 卡 BI 150（共 8 芯）
并行设置	TP=8，PP=4，DP=1，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，启用计算通信重叠
算网协同设置	使用异属算力，使用算网协同，使用异构算力芯片

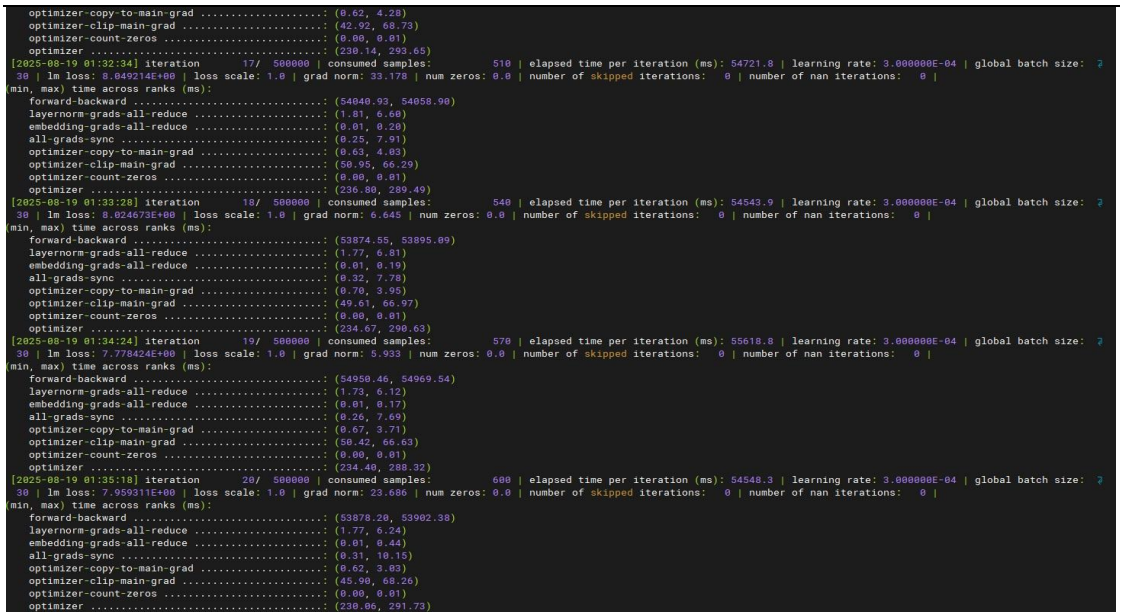

图 46 训练任务 11 的日志

表 18 训练任务 11 的训练性能数据

每迭代完成时间	54.5 秒
TGS（Tokens/gpu/s）	322.10
Samples/s	0.5505

测试结果分析：1、调度系统可调度异构算力芯片协同完成一个训练任务；2、增加算力芯片总数后，每迭代完成时间随之下降，同时每秒处理的训练样本数也随之增加；3、对比训练任务 1 的 TGS 指标，此时训练任务 11 的 H20+BI150 混合训练效率达到 95.47%。

5.2.6 基于算网协同的多流水线跨域训练能力验证

向调度系统提交训练任务 12，如表 19 所示，等待调度系统的反馈。选择在集群 1 和集群 3 运行训练任务 12，最后观察训练任务 12 的日志，如图 47 所示。

表 19　训练任务 12 的描述信息

大模型	Mixtral 8x7B，总层数 56 层，总参数量 80B
GPU 数量	32 卡 H20
并行设置	TP=4，PP=4，DP=2，EP=1
其他训练设置	sequence length=16384, micro batch size=2, global batch size=30, 浮点数精度 fp16，启用计算通信重叠
算网协同设置	使用异属算力，使用算网协同，不使用异构算力芯片

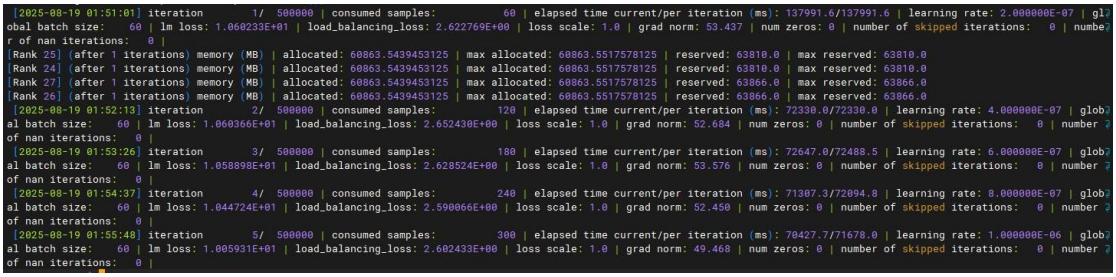

图 47　训练任务 12 的日志

测试结果分析：在启用算网协同、计算通信重叠技术后，用户仍然可以进行多流水线跨域训练，从而可利用更多的算力资源来协同完成一个训练任务。

六、 总结与展望

大模型技术的演进正经历从通用领域向行业场景的深度渗透，这一转变直接推动训练需求从大规模预训练向精细化后训练、场景化微调延伸。在此过程中，业界的技术焦点逐渐从单一数据中心的性能优化，转向跨地域、异构环境下的协同训练能力构建。然而，随着多行

业大模型落地进程的加速，不同主体间因资源权属、管理策略、安全规范差异形成的 “异属壁垒”，正成为比技术异构性更突出的制约因素 —— 数据主权保护、资源调度权限分割、跨主体信任机制缺失等问题，使得跨域算力池化的难度远超单纯的技术适配挑战。

白皮书提出的大模型跨域训练池化调度技术体系，以破解算力资源的“异属异构异地”三大核心挑战为目标，构建了覆盖业务层、管控层、资源层的全栈式技术架构。在业务层，通过动态任务拆分与子作业协同机制，实现大模型训练任务对跨域资源的弹性适配；在管控层，依托多异属队列协作、联合抢占等策略，解决不同主体资源的统一调度与优先级协同问题；在资源层，借助跨主体 RDMA 网络虚拟化等技术，突破网络安全隔离与高性能通信的矛盾，保障跨域数据交互效率。这一体系不仅实现了跨地域、跨主体异构算力的高效整合与动态协同，更在提升资源利用率、缓解高端 GPU 供给压力的同时，为大规模分布式模型训练提供了从任务发起至资源释放的全生命周期技术支撑。

后续，大模型跨域训练池化调度技术体系将持续以“全国一台计算机”为目标愿景，实现算力泛在化、效率本地化与生态开放化。通过全域算力池化整合不同地域、主体的异构计算资源，形成统一供给平台，让用户按需取用如同用电般便捷；打破物理与逻辑层面的多重边界，消除通信壁垒并屏蔽硬件、软件、权属差异，使任务运行如单一集群；在全域协同中借助智能调度、网络优化等保持接近本地集群的训练效率；同时构建开放生态，支持多主体接入，实现技术共享、

标准共建，最终实现多方利益共赢。

算力泛在化。实现算力资源的 “无处不在、按需取用”。通过全域算力池化技术，将分散在不同地域、分属不同主体的 GPU、NPU、CPU 等异构计算资源整合为逻辑上的统一算力供给平台。用户无需关注资源的物理位置、硬件型号或权属归属，只需通过标准化接口提交训练任务，系统即可自动匹配最优算力资源，实现“像用电一样用算力”的便捷体验。

效率本地化。在全域协同中保持接近本地集群的训练效率。通过智能调度策略将计算任务分配至距离数据源头最近的算力节点，减少跨域数据传输量；利用网络感知的通信优化技术（如动态压缩、路径选择）降低长距离通信延迟；结合异构硬件特性动态调整计算精度与并行策略，使跨域训练的吞吐量、收敛速度接近同构本地集群水平。

生态开放化。构建多方共赢的技术生态体系。通过开源框架、标准化接口与模块化设计，支持硬件厂商（如国产 GPU 厂商）、软件开发者（如训练框架团队）、算力提供方（如数据中心）、用户（如 AI 企业）等多主体接入。硬件厂商可通过统一抽象层快速适配主流训练场景，开发者可基于标准化接口扩展新功能，用户则能在兼容多类型资源的环境中灵活选择方案，形成“技术共享、标准共建、利益共赢”的开放生态。