HUAWEI | CAICT 中国信通院 | | 4Paradigm | HealthLink 泛联康健 | 复旦大学

云原生AI技术架构白皮书

白皮书编制组

华为云计算技术有限公司

叶坤奇　张　琦　张永明　蔡智源　王雷博　魏鹏程　陶　希　陈佳敦　朱佳玮　马红伟
左鹏飞　付森波　张超盟　范恒龙　鲍　玥　冯绍宝　朱　磊

中国信息通信研究院云计算与大数据研究所

刘如明　杜　岚

行吟信息科技（上海）有限公司

徐瑞文　胡伟琪　余　奕　陈　磊　熊　峰

第四范式（北京）技术有限公司

李孟轩

远盟康健科技有限公司

杨　宇　陈　浩

复旦大学

彭　鑫　沈立炜　陈碧欢

目录 | CONTENTS

01 背景和前言

1.1 大模型开创智能时代的新纪元，AI 产业迎来新一轮创新浪潮……02

1.2 云原生助力 AI 产业突破发展瓶颈，云原生 AI 成为产业发展新范式……02

02 云原生 AI 基础设施发展和挑战

2.1 云原生 AI 技术的演进……05

2.2 算力诉求井喷，AI 产业面临挑战……06

03 云原生 AI 技术概论

3.1 云原生 AI 资源管理系统建设要点……09

3.2 云原生 AI 训练系统建设要点……15

3.3 云原生 AI 推理系统建设要点……26

3.4 云原生 AI 边缘云系统建设要点……30

3.5 弹性伸缩，应对 AI 任务浪涌挑战……32

04 云原生 AI 技术应用

4.1 云原生 AI 跨地域多集群协同……38

4.2 云原生 AI 算力效能优化……41

4.3 云原生 AI 云边协同计算……46

4.4 大模型云原生化解决方案……49

4.5 云原生 AI 设备驱动管理……51

05 云原生 AI 行业实践

5.1 社交平台 RB 云原生 AI 平台应用加速实践……54

5.2 AI 解决方案提供商 FP 多场景 AI 云原生化实践……58

5.3 医疗科技公司 HL 云原生 AI 智能医疗实践……60

PART 01

背景和前言

1.1 大模型开创智能时代的新纪元，AI 产业迎来新一轮创新浪潮

1.2 云原生助力 AI 产业突破发展瓶颈，云原生 AI 成为产业发展新范式

1.1 大模型开创智能时代的新纪元，AI 产业迎来新一轮创新浪潮

AI 软件及应用市场持续增长，AI 大模型成为产业主要增长点。据 IDC 估计，2026 年中国人工智能软件及应用市场规模将达到 211 亿美元，各行业的 AI 需求极大地推动着 AI 市场增长。随着数字经济、元宇宙等概念的逐渐兴起，人工智能进入大规模落地应用的关键时期 , 但其开发门槛高、应用场景复杂多样、对场景标注数据依赖等问题开始显露，阻碍了规模化落地。以 ChatGPT 为代表的 AI 大模型的横空出世改变了这一局面。凭借其优越的泛化性、通用性、迁移性，AI 大模型为人工智能大规模落地带来新的希望。面对人工智能的各种挑战，AI 大模型的出现提供了通用化解决方案，从无标注数据中通过自监督学习获取大量“知识”，实现用更统一的方式推动人工智能产业落地。

广泛智能需求驱动 AI 产业不断创新，大模型助力各行业生产力变革。随着办公、制造、金融、医疗、政务等场景中降本增效、生产自动化、降低风险、提高诊断准确率、提高政务服务效率等多方面的 AI 智能需求，AI 产业迎来了井喷式的创新和发展。凭借在文字、语音、图像、视频等多模态处理能力上的跃迁，AI 大模型摇身变为“助理”、“专家”走入办公室、制造车间、金融市场、医疗机构、政务大厅，结合传统软件使得各个行业更加智能化、自动化。AI 大模型已然改变了我们的生活和工作的方方面面，成为各个行业不可或缺的重要助手。

1.2 云原生助力 AI 产业突破发展瓶颈，云原生 AI 成为产业发展新范式

AI 产业面临数据、算法、算力等多方面发展瓶颈。据 IDC 统计 , 中国数据规模将从 2021 年的 18.51ZB 增长至 2026 年的 56.16ZB，年均增长速度 CAGR 为 24.9%，增速位居全球第一。随着数据量的高速增长，数据特征高维、模态格式多样的趋势也逐渐明显，对数据的 AI 建模也相应地更加复杂，计算复杂度会随之呈指数增加，数据标注难度也会增加。同时，海量的数据将不可避免带来更大的数据噪声问题、数据偏见风险。与此同时，AI 应用场景更加多元化、复杂化，往往需要对多个任务进行深度融合和统一建模，这意味着厂商需要针对不同场景、不同任务开发大量的算法和模型，增加了 AI 应用的开发难度。算力方面，需要针对不同的场景和高性能计算能力进行拓展融合 , 满足研发企业的多芯部署、分布式优化、高性能计算等需求，这涉及了计算资源的灵活调度和统一运营管理，给企业 AI 创新带来了额外的成本。

云原生 AI 成为 AI 产业发展的新范式。为了突破 AI 产业的发展瓶颈，云原生 AI 技术应运而生。一方面，云原生技术为 AI 应用运行提供了一个可扩展、高可靠的平台，更好地支持 AI 开发和使用。目前，基于 Kubernetes 的云原生可以有效管理各类网络、存储和计算资源，已逐步演变为实际上的云操作系统，服务

于私有云、公有云以及混合云环境。基于其高可用特性，云原生系统可通过自动故障恢复机制在故障发生时迅速恢复服务，确保 AI 应用的稳定运行。其次，利用 Kubernetes 自动伸缩功能带来的出色扩展性，云原生可以根据 AI 应用需求快速增加或减少计算资源，满足不同场景下的计算需求。同时，云原生具备良好的兼容性，可以与各种 AI 框架和工具无缝集成，实现 AI 应用的快速开发和部署。此外，云原生提供了丰富的计算（如 CPU 和 GPU）、网络和存储能力，并提供隔离和受控共享机制，加速了 AI 应用开发的效率和性能，并降低了企业的成本。另一方面，AI 也可以从调度资源、安全等方面增强云原生。在涉及多个优化标准的情况下，AI 可以分析集群的历史使用情况并预测未来工作负载模式和资源可用性，更好地调度云基础设施资源，进而降低能源消耗和使用成本。在安全方面，AI 可以分析大规模数据集并预测系统中的潜在威胁或弱点。用于检测异常网络行为的 AI 模型可以轻松地用于保护工作负载或在边缘部署中的一组集群，加强企业对新兴网络威胁的防御。

本白皮书重点关注云原生 AI 基础设施层支持 AI 开发和使用，结合云原生开源生态发展现状和行业实践，深入分析云原生 AI 技术落地所面临的技术挑战并给出具体的技术指导方案。

PART 02

云原生 AI 基础设施发展和挑战

2.1 云原生 AI 技术的演进

2.2 算力诉求井喷，AI 产业面临挑战

云原生技术本质上是基础设施云化和与之配套的服务（例如 CI/CD 就是如何在云化的基础设施部署软件）的技术。这在云原生 AI 里也是一样的，云原生 AI 基础设施是云原生 AI 技术最为基础的一环。云原生 AI 基础设施向上为 AI 训练作业、推理服务及模型开发等各类 AI 业务提供任务编排和调度能力，向下对多数据中心的异构硬件设备统一纳管并提供高效、可靠的资源供应能力。这一章将简短地回顾一下云原生 AI 基础设施的技术演变历程，我们会看到如今云原生 AI 技术面临的挑战的来源。

2.1 云原生 AI 基础设施的演进

2018 年图灵奖获得者计算机体系结构泰斗约翰 · 轩尼诗 (John Hennessy) 和戴维 · 帕特森 (David Patterson)，在颁奖典礼上发表了题为“计算机体系结构的新黄金时代”(A New Golden Age for computer Architecture) 的演讲[①]，指出摩尔定律 (Moore’s Law) 和登纳德定律 (Dennard Scaling Law) 走到了尽头，处理器的晶体管密度和单位面积功耗已接近极限，处理器的性能提升不再遵循摩尔定律，后摩尔定律时代到来。

AI 技术的发展和新的软硬件接口抽象为云原生基础设施带来了新的挑战和机遇，以面向特定领域体系结构 (Domain-Specific Architecture, DSA) 处理器为代表的新架构能够提供更高的性能，更低的成本和更优的能效。

2022 年 11 月 30 日 OpenAI 公司推出了智能聊天机器人 ChatGPT，在发布后的 2 个月内用户数量就突破 1 亿，成为史上用户增长最快速的现象级应用。ChatGPT 表现出的对文本的超凡理解力和生成能力，让工业界对 AGI 从学术研究走进实际的商业应用有了前所未有的信心，各类基于 Transformer 架构的 AIGC 大模型应用如雨后春笋，国内也出现了百模大战的态势，更进一步出现了 Stable Diffusion 和 Sora 等多模态大模型。在近几年的大模型研究和工程实践中，业界发现模型的训练数据、参数量和计算量越大，模型的效果越好，模型规模与模型效果呈现显著的正相关，虽然学术界存在争议，但大模型的 Scaling Law 仍然是业界的基本共识。

为应对大模型对算力、存储（带宽、容量）需求，必须把大量加速卡和服务器节点通过高速总线和网络连接起来，利用节点内总线（Scale-Up）和节点间网络（Scale-Out）的层次化扩展能力，构建大规模 AI 集群以提供充足的算力供应，随着模型尺寸的持续增长，AI 集群的规模也越来越大。典型的 AI 集群具有两个或三个网络平面及一个高速总线平面，分别是：前端网络平面，用于集群管理和 AI 作业的调度发放；后端网络（Scale-out 或 Back-end）平面，用于扩展多 AI 服务器节点，通过高性能网络 Infiniband 或以太网

① https://www.jiqizhixin.com/articles/2019-01-30-12

把不同节点的 GPU/NPU 卡通过 RDMA 协议连通起来，主要用于模型参数的数据同步（注：也有厂商称之为参数平面）；存储网络，通过专用的存储网卡和交换机将训练节点和存储设备连接起来，用于训练数据读取和模型快照（Checkpoint）存取；高速总线（Scale-Up link）平面，通过高带宽高可靠的片间总线（如：PCIe/NVlink 等）将节点内加速卡互联起来，用于大模型训推过程中的梯度更新等数据同步。

2.2 算力诉求井喷，AI 产业面临挑战

OpenAI/Meta/ 字节跳动等公司近期所披露出的 AI 集群的规模都超过万卡，在他们的研究报告和相关的学术论文中提出大量当前 AI 业务在使用大规模算力集群过程中遇到的挑战和问题，这里我们列举几个核心问题：

线性度问题

相对于单卡和单计算节点的计算效率，AI 计算任务在多卡多节点上的执行是否能够达到线性的收益目标，特别是随着集群规模的扩展，线性度能够持续保持。以模型训练为例，模型训练的吞吐（样本数 / 秒）= 单卡训练吞吐（样本数 / 秒）* 加速卡数量 * 线性度，理想的线性度是趋近于 1。

通过高性能总线将多个节点的加速卡连接起来的超节点（SuperPOD），打破了传统节点的模型，如英伟达 DGX H100 支持将 32 个节点的 256 个 GPU 组成一个超节点，超节点内的 GPU HBM 和 CPU 内存统一编址，支持更大参数规模的模型加载。这超出了传统节点资源和拓扑模型的表达能力。

而在 Scale-Out 扩展方面，一般采用二层或三层 Spine-leaf 拓扑模型，通过无带宽的收敛 InfiniBand 或以太网络将加速卡节点连接成 AI 集群。要保持 AI 算力集群中 AI 任务的线性度，需要综合作业节点间的网络拓扑和 AI 任务的并行策略及其通讯需求进行作业任务的层次化调度，这对集群的调度器提出了新的要求，即：要感知集群的资源的网络拓扑和（超）节点拓扑，并根据 AI 任务的并行模式和通讯要求，将任务切分并调度到合适的节点和卡上，目前云原生 AI 调度器方案在拓扑感知及作业并行策略表达及调度算法方面存在明显的能力缺口。

大模型训练的主要并行模式和通信需求如下，通信模式具有显著特征：

1. **周期性强，每轮迭代的通信模式一致；**
2. **流数量少，单流带宽大，同步突发。**
3. **通信量大，带宽需求高。**

并行模式	特 征	通信需求	
Tensor 并行 (TP)	通信量巨大（百 GB），通信时间不可掩盖	节点内 allreduce	超高带宽
Pipeline 并行 (PP)	通信量较大（模型相关，百 M-GB 级），通信时间不可掩盖 / 流水可掩盖	跨节点 P2P	中带宽
数据并行 (DP)	通信量大（GB 级），通信时间计算可大部分掩盖	跨节点 allreduce	高带宽
MOE 并行	通信量大，通信时间不可掩盖	跨节点 alltoall/allreduce	高带宽

表 2-2-1　大模型并行模式和通信需求

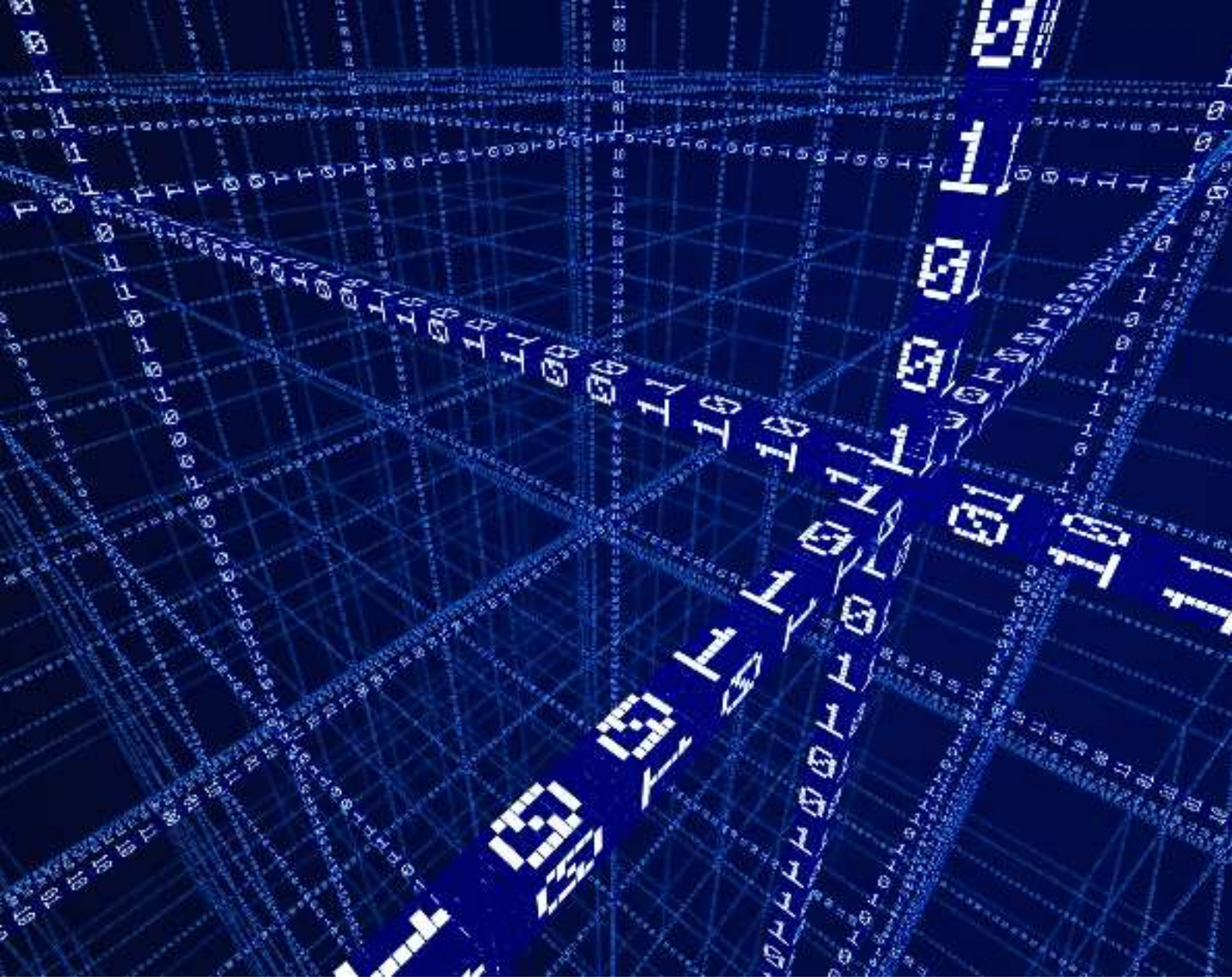
集群可用度和资源利用率问题：

是 AI 集群使用者和供应者共同关注的问题，集群的可用度直接关系到 AI 任务能否在预期的时间内完成，而可用度和资源利用率对企业内的 AI 基础设施部门或公有云厂商则意味着服务 SLO 能否达成，能否通过压低 AI 集群的资源成本取得盈利

AI 大模型支持通过保存快照（CheckPoint）加速故障恢复，避免训练进度丢失。提升 Checkpoint 的存取性能，及时发现故障并快速恢复都有待云原生 AI 中的存储、故障和检测恢复组件的提供更加完备和高效的方案。

考虑到集群内运行不同租户（公有云）、不同规格、不同运行时长的 AI 任务，容易产生资源碎片，需要能够平衡集群资源利用率和 AI 任务性能目标，这要求云原生 AI 的重调度和快速任务迁移协同解决。

对于 AI 开发者而言，AI 基础设施首先应该屏蔽底层基础设施的细节，使 AI 开发者可以聚焦在数据质量的提升和模型架构的优化。加速卡有不同的型号和参数、加速卡间通过不同的网络拓扑通信，不同的网络平面也有各自的带宽限制，如果 AI 任务部署前还需要考虑这些硬件因素，一方面增加了 AI 开发者的学习成本，另一方面也会耗费他们额外的精力，降低 AI 开发者的产出效率。



PART 03

云原生 AI 技术概论

3.1 云原生 AI 资源管理系统建设要点

3.2 云原生 AI 训练系统建设要点

3.3 云原生 AI 推理系统建设要点

3.4 云原生 AI 边缘云系统建设要点

3.5 弹性伸缩，应对 AI 任务浪涌挑战

如前文所述，云原生 AI 技术包含了很多方面，从底层的硬件和数据中心，到容器集群管理，编排调度系统，再到上层的云原生 AI 应用。而云原生 AI 由于技术较新，很多企业在构建云原生 AI 的时候仍面临很多问题，本章将对现今云原生 AI 面临的热点技术问题，给出前沿的技术指导。

图 3-1-1　云原生 AI 技术架构

3.1 云原生 AI 资源管理系统建设要点

云原生 AI 资源管理系统涵盖了 AI 资源管理、矩阵算力基础设置管理、云原生资源管理、资源画像、垂直弹性、水平弹性以及智能 HPA（Horizontal Pod Autoscaling）等多个方面，它们共同构建了一个灵活、高效、智能的 AI 资源调度与管理框架，是驱动现代企业和组织智能化转型的核心动力。

1、现状与问题

AI 算力资源发展至今，从传统的 CPU 到 GPU，再到百家齐鸣的 NPU、TPU、DPU 等等，AI 云计算已经进入了一个高速发展的 XPU 时代。在 AI 算力业务蓬勃发展的时代背景下，AI 算力诉求急剧膨胀，从最开始的单机单卡、单机多卡，到现在的千卡、万卡集群，这也引出了一系列的问题和挑战：

集群规模快速膨胀，AI 资源管理复杂度上升。

随着 AI 产品的大众化、规模化，搭载商业级算力芯片的大规模算力集群，成为了各个科技型企业的必

备武器，AI 算力集群规模也日益膨胀，这就带了不可避免的问题：如何能更高效的管理成千上万的 AI 算力资源。

AI 芯片种类繁多，对于 AI 资源管理的可扩展性有了更高要求。

无论是现今一家独大的英伟达，还是厚积薄发的华为、谷歌、AMD，都在推出 AI 场景算力芯片，例如英伟达的 GPU、华为的昇腾 NPU 及谷歌的 TPU。AI 算力云厂商或是 AI 型企业，面对各家算力厂商迥异的架构，也急需有一套可扩展性更好的 AI 资源管理架构。

参数面网络等新型 AI 资源，对于 AI 资源管理提出了新的挑战。

大模型、自动驾驶、AIGC 的横空出世，大规模的算力参数面互访网络成为了必需品，参数面网络提供的超高带宽，发展出了计算机超节点架构，计算机超节点是一个由多个和多种计算 (CPU/NPU)，内存，IO 设备等计算机资源单元，高速互联紧耦合在一起的集群计算系统，是生成式 AI 时代的产物。区别于传统以服务器中心松耦合架构，超节点是去中心化的紧耦合架构。随着技术的进一步演进，未来超节点内所有服务器的设备可做到灵活组合成为各种算力单元，也可被称为矩阵式算力。为了能够有效利用超节点内的资源，相关联的算力参数面网络设备及其拓扑的管理，也就成为了 AI 算力资源管理的新课题。

XPU 计算吞吐能力快速提升，I/O 瓶颈越发严重。

当前 CPU 与 XPU 的发展出现严重错位；XPU 的算力虽然远超 CPU，CPU 拥有的内存容量是 XPU 无法比拟的；这就导致在海量数据训练的过程中，数据不得不同时分布在 CPU 和 XPU 的内存上，而为了最大限度发挥 XPU 算力的效率，数据必须能够尽快的被 XPU 访问到而不是浪费时间等待数据；随着 AI 大模型参数的指数级增长，AI 大模型训练和推理越来越面临内存墙和 IO 传输墙的挑战，即 XPU 内存容量和 IO 带宽的增长跟不上 AI 模型大小的增长速度。因此我们需要构建可扩展的数据管道以高效地将数据传输到 XPU 计算设备至关重要。在云上多租业务场景下，我们需要注意并确保 I/O 瓶颈不会出现在重要业务场景比如对性能要求极高的推理场景。

Google[②]和 Microsoft[③]的研究表明，高达 70% 的模型训练时间被 I/O 占用。字节跳动在 MegaScale[④]

② Jayashree Mohan, Amar Phanishayee, Ashish Raniwala, and Vijay Chidambaram. 2021. Analyzing and mitigating data stalls in DNN training. Proc. VLDB Endow. 14, 5 (January 2021), 771–784. https://doi.org/10.14778/3446095.3446100

③ Derek G. Murray, Jirí simsa, Ana Klimovic, and Ihor Indyk. 2021. Tf.data: a machine learning data processing framework. Proc. VLDB Endow. 14, 12 (July 2021), 2945–2958. https://doi.org/10.14778/3476311.3476374

④ Jiang, Ziheng, et al. "MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs." arXiv preprint arXiv:2402.15627 (2024).

的最新研究表明可以达到 55.2% 的模型 FLOP 利用率，换句话说，XPUs 有接近一半的时间的时间处于闲置状态，造成的大量的时间和金钱的浪费。现在，由于 AI 大模型需要的算力和数据实在过大，很多中小型 AI 厂商无法获得足够的计算和存储资源来训练和优化模型，所以不得不在云数据中心购买资源。在云上，数据分布在不同的地理位置，数据集大小已经远远超出了本地和单个 XPU 存储容量，云上 AI 训练和推理已经成为了新的范式。云上 AI 训练和推理过程涉及移动数据集或复制数据到 XPU 设备上，为了最大限度地提升 AI 场景 XPU 效率，I/O 优化是云基础设施的重要环节。

随着 AI 计算规模增大，例如大规模 AI 训练，需要多卡甚至多个节点同时参与一个任务的计算，其中一个关键点就是如何支持节点内和节点间算力的高速通信，以便他们可以作为一个巨大的加速器相互协作。

2、AI 通用算力基础设施关键技术和价值

面对以上的问题和挑战，作为 AI 云原生计算基座，kubernetes 的社区提供了针对 AI 资源设备的管理机制，Device Plugin 模式和 Dynamic Resource Allocation 模式。

（1）大规模设备管理

无论集群的 AI 算力规模如何膨胀，AI 算力设备终究需要依托在服务器节点上，这形成了一对多的关系，对于 AI 算力的管理也就演变成了针对算力节点的管理。kubernetes 社区针对拥有 AI 算力资源的节点，推出了 Device Plugin 的插件框架（DP 模式），在每一个算力节点上运行 Device Plugin 进程。各个 Device Plugin 进程仅需要管理自身节点上的少量 AI 算力设备，并将可用算力设备数上报至集群数据中心侧，由 kubernetes 的资源调度系统进行后续的调度使用。

（2）设备管理的可扩展性

Device Plugin 模式，将设备管理抽象成了若干管理 API，包括：监听 (ListAndWatch)、分配 (Allocate) 等等，kubernetes 以 GRPC 协议，在节点上与 Device Plugin 进程进行通信。所以 Device Plugin 进程自身的逻辑和 kubernetes 是解耦合的，算力厂商、第三方使用者仅需要实现极简的 Device Plugin 管理 API，进行各种异构算力芯片的扩展对接。

（3）新型 AI 设备的管理

Device Plugin 模式满足了绝大多数的基本算力管理场景，但对于一些新型的复杂 AI 设备，仍然有所欠缺，例如大规模模型训练场景的高性能网络设备 (RDMA 设备)，是依赖于网络拓扑感知的，就近分配资源才能避免长距离网络访问，避免 AI 训练效率降低。

而 Device Plugin 模式，存在上报资源格式单一、管理模式单一等缺陷，无法满足复杂的 AI 资源设备管理场景。针对这些问题，Kubernetes 社区提出了动态资源分配 (DRA: Dynamic Resource Allocation) 的模式，其有以下功能特点：

支持自定义资源参数：DRA 的资源分配管理采取 CustomResource(CR) 的方式，CR 其高度可扩展的特征，允许开发者进行特殊 AI 设备的参数扩展；

支持设备的初始化和清理过程：设备的申请 / 注销是由中心侧控制器负责，Kubelet Plugin 则负责响应，进行设备的初始化 / 重置，这个过程是与 Pod 的生命周期解耦的。这个机制允许我们对于 AI 设备进行一些初始化操作；

支持设备的部分分配：相较于 Device Plugin 的独占分配，DRA 支持通过 ResourceClaim 的方式，让设备在多个 Pod 或容器间的动态共享。

相较于 Device Plugin 模式，DRA 有更加丰富的语义，可以满足更复杂的设备管理场景，但 DRA 带来了丰富语义和扩展性的同时，其管理成本的开销、效率也是有所增加，所以 DRA 的出现并不代表替换 Device Plugin，在一些传统的 AI 设备管理场景，Device Plugin 仍然是第一选择。

3、矩阵算力基础设施关键技术与价值

面对问题和挑战，作为 AI 云原生基础设施资源底座，kubernetes 构建了面向超节点架构的整套资源管理方案。

虽然计算机超节点的 High-Speed Link 高速互联能够提供比传统互联更高的带宽，但单路径带宽仍无法匹配计算单元的吞吐，基础设施层通过构建全局多路径 I/O 加速技术，大幅提升了节点内与节点间 I/O 性能。

为匹配 AI 行业所需的庞大算力需求，基础设施硬件从主从架构逐步演进至对等架构，传统的资源管理模型不再适用，需要构建面向对等架构的资源管理模型，实现资源的高效管理与合理配置。

（1）全局多路径 I/O 加速技术

通过对比各代 GPU 的 GPU 算力和 CPU-GPU IO 带宽，不难发现传输墙的限制正在加剧，短期内不太可能得到解决。PCIe 带宽非常有限，PCIe Gen3 的理论带宽是 32GB/s，PCIe Gen4 的理论带宽是 64GB/s，而实测带宽大概分别是 24GB/s 和 48GB/s。在 AI 训练中，每完成一轮计算，都要同步更新一次参数，模型规模越大，参数规模一般也会更大，这样 GPU 之间通信（P2P）能力对计算效率影响就比较大。NVLink 的

目标是突破 PCIe 接口的带宽瓶颈，提高 GPU 之间交换数据的效率。基于定制互连 NVLink，GPU 到 GPU 的带宽明显快于 PCIe 带宽。另外，GPU 显存带宽（HBM、GDDR）是大模型推理的性能瓶颈。而 GPU 到 CPU 之间的互连（PCIe）是瓶颈。典型的 x86 CPU 只能通过 PCIe 与 GPU 通信，而 NVLink-C2C 的带宽远超 PCIe 并具有缓存一致性的优势。目前在 GH200 和 GB200 等超级计算机中，NVLink 并开始应用于 GPU 服务器之间的互连，进一步扩大 GPU（以及其显存）集群的规模。

全局多路径加速是指我们可以利用单机内 CPU 与 GPU 等不同芯片的多路径，以及跨主机的多路径提升 I/O 带宽，缩短数据的传输总时延；GLake 多路径通过利用 NvLink 和同一节点上的多个 PCIe 和 NVLink 路径来加速 CPU-GPU IO 传输。但这远远不够， 大语言模型（LLM）对显存容量的需求非常迫切，巨大的显存容量符合大模型的发展趋势。那么，这个前所未见的容量是通过大规模的机器互联来实现；在这种更大规模互联的场景下，集群内跨设备之间的 I/O 通信可以采用的路径也会越来越多，包括 CPU 与 CPU，CPU 与 XPU，XPU 与 XPU 之间的高速互联；

在未来超节点架构中，一个物理超节点是由很多计算和存储设备通过网络连接起来的集群；一个物理超节点可能会被拆分成多个逻辑节点分配给不同租户的不同业务，这些业务可能会同时分布在多个计算单元上，这些计算不同业务对 I/O 的需求模式可能有区别。两个互相通信的设备之间可能存在不同时延，不同带宽的多条路径。如果将大量的 I/O-Intensive 负载放置到同一个物理设备上怎么样来动态选择 IO 传输路径，保证带宽的高效利用同时又能满足不同业务的 SLO 呢？有如下几个解决方法：

路径规划：通过劫持底层 I/O 通信算子，并能够在跨集群场景下分布式劫持通信需求，结合全局拓扑，针对当前不同设备之间的通信需求快速地生成对应的可用路径；并根据每一条路径上的当前带宽和可用性规划传输的数据量大小并按需使用对应路径；在数据量传输较大的场景还可以提高性能——通过同时使用多个数据路径，AI 应用程序获得更高的数据传输吞吐。I/O 同时通过多条路径传输提高性能并降低延迟；

我们不仅仅可以通过 XPU 之间的 High-Speed Link 还可以结合通用计算设备之间的 High-Speed Link 进行 I/O 加速，从而充分利用构建起来的高速互联通道，实现价值最大化；

感知业务特征和集群拓扑：通过持续在线监控和避免 I/O 负载瓶颈来简化多路径管理。当某一条路径 I/O 带宽接近极限的时候或者发生故障的时候，能够通过其他路径来完成 I/O 任务，保证在多设备并行计算和数据传输过程中实现最佳性能和提升业务可用性；当路径过于饱和的时候，可以根据业务优先级分配 I/O 带宽，优先满足时延敏感型业务的 I/O 带宽需求，避免关键业务性能受损。

减少非必要数据传输：利用高性能缓存层或者在 XPU 可快速访存的内存中缓存常用的训练数据来加速 I/O。与此同时也可以通过冷热数据分级分区存放，热数据尽量放在 I/O 访问时延较低的内存介质中，冷数据放在 I/O 访问较远的内存介质中，从而尽可能降低非必要的数据加载；在 AI 计算过程中，充分利用好多个 XPU 之间并行，并通过数据分片和调整 mini-batch size 来更好地利用好 XPU；

（2）超节点资源管理模型

传统资源管理模型的基本算力单元为单台服务器，服务器模型内包含各种设备（CPU, 内存以及 I/O 设备等），资源池模型由服务器模型聚合而成，其资源分配也是以服务器为基本粒度，云化场景下的云服务器也仅是设备数量存在差异，其基本建模均保持一致。超节点为去中心化的架构，虽然物理设备仍依托于服务器之上，但超节点内配备有超高速互联网络，其内所有设备均可以灵活组合成不同的算力单元，超节点架构基本算力单元不再是单台服务器，传统资源管理模型已不再适用。面向超节点架构，Google 的 TPU 服务构建的层次化的资源管理模型，是业界当前比较成熟的解决方案。

超节点资源管理模型与资源切片：超节点资源管理模型包含三个基本算力单元模型：XPU、CPU 和内存，其他设备均建模为附属模型。在资源管理模型中将基本模型又被抽象为资源节点 Node，超节点的高速互联被抽象为连接资源节点之间边 Edge，一个超节点被抽象为一个 SuperPoD，多个 SuperPoD 组成一个集群 Cluster，资源池就是集群的聚合。

SuperPoD 的资源分配模型是 XPU、CPU 和内存的组合，称为超节点资源切片 slice。其中 XPU 的资源分配粒度为设备，CPU 为 CPU Core，内存为容量。Edge 作为资源组合的约束，对资源的组合形式进行限制。比如客户申请一个 64XPU，320CPU Core, 1024GB 内存的 slice，超节点资源调度器不仅要调度足量的 XPU、CPU 和内存资源，还要通过图匹配算法确保被调度的资源节点之间存在直连 Edge。基本算力单元之外的设备不参与资源调度过程，而是通过规格预定义的方式进行管理，在 AI 场景下这些设备的分配量一般与 XPU 资源量锚定，按照不同的 XPU 请求量划分为若干档位。

超节点资源拓扑感知： AI 业务场景下所需的通信量非常大，其通信算法都会根据基础设施网络拓扑进行编排优化，以达到充分利用网络带宽的目的。为了有效利用超节点的高速互联网络，客户也需要感知到超节点内部的拓扑结构来优化通信算法。然而算力服务提供商出于安全和保密方面的考虑，一般不会对客户暴露物理信息，而是通过抽象方式隐藏物理信息。AWS 提供了一套网络拓扑的抽象建模思路能够在满足通信算法优化需求的同时隐藏物理信息。超节点资源拓扑感知模型将不同的网络设备抽象为虚拟的网络节点 NN（network node），并为每一个 NN 进行逻辑编号，如 NN001，NN002。客户在查询超节点 slice 的设备拓扑时，接口会返回每一个设备所属的每个层级的 NN，客户可以根据 NN 的逻辑编号是否相同来确定

设备间高速互联的拓扑结构。

超节点资源高可用：高可用能力是大规模集群系统必须具备的基本能力，基础设施层的高可用能力之一是故障设备替换。故障设备替换指的当客户正在使用的设备出现故障时，使用一个正常设备将其替换掉，帮助客户快速恢复业务。在超节点架构下，由于超节点内的设备之间具备高速互联网络，所以可用于替换的设备必须在超节点内部，不能跨超节点进行设备替换。在超节点架构下执行故障设备替换时，资源管理平台会约束调度系统的调度范围不能超出设备所在的超节点。此外，由于超节点规模有限，为了确保超节点内存在可用于替换的设备，资源管理平台会在每个超节点内预留部分设备作为保底手段。在故障替换时会优先选择非预留的空闲设备，在非预留空闲设备不满足替换需求时才会动用预留资源。在某个预留设备被使用后，预留设备池的容量随之减少，资源管理平台会周期性的扫描超节点内设备使用状态，若存在被释放的设备则将其加入预留池，以实现预留池容量的轮转。同时，资源你管理平台也会通知运维人员及时维修故障设备。在 AI 场景下，为了与 Checkpiont 机制相配合，资源管理平台会对外暴露设备替换接口。AI 作业管理平台在保存好现场后调用此接口进行故障设备替换，替换成功后再通过读取 checkpoint 恢复业务 。

除设备故障外，网络断连也是典型的故障场景，超节点资源管理平台采用借轨通信的方案解决此类问题。借轨通信是指在设备 A 与 C 的当前互联路径中断的情况下，由于设备 A 和 C 仍然与设备 B 保持通信连接，设备 A 可选择从设备 B 跳转的方式与设备 C 实现通信。跳转节点通过路径规格算法进行优选。

3.2 云原生 AI 训练系统建设要点

云原生 AI 训练能力集成了 AI 调度加速、AI 训练存储加速、AI serverless 训练以及 AI 故障自愈等多项关键功能。这些能力不仅极大地提升了 AI 模型训练的效率和性能，也为企业的智能化转型提供了强有力的支持。

1、AI 调度加速

（1）现状与问题

在训练阶段，通过大量数据和算法，AI 模型学会识别和生成规律。有别于一般的通用业务场景，AI 大

模型训练对数据传输的带宽和性能有更高的要求。同时，随着大模型的出现，AI 训练 / 推理任务不再是单卡或单机的规模，通常表现为多个容器实例组成的分布式任务负载，由此对 AI 任务智能调度能力提出了更多挑战。

分布式调度死锁和忙等：一个分布式训练作业通常由一批相关联的 Pod 联合执行训练任务，比如 TensorFlow 中的一组 PS 和 Worker，这些 Pod 会用到相同的资源执行相同的任务，并且通常都是同时起且同时停。当集群中并发提交多个训练作业，在资源有限的情况下，每个训练作业仅部分 Pod 被成功调度从而获取到集群资源，此时各训练作业均在等待更多资源以满足作业运行的条件，造成作业之间的资源死锁和忙等，训练作业无法有效执行。

算力高效利用：在大模型任务训练场景，动辄需要几百甚至几千张 GPU 卡的算力，服务器节点多、跨服务器通信需求巨大，处于同一个机架、Tor（Top of Rack）或者超节点内的节点之间通信效率各不相同，同一个超节点网络拓扑内卡之间的网络通信最高，Tor 内通信效率次之，最后为普通节点之间通信。在传统的分布式并行策略中，Tensor 并行（Tensor Parallelism）和流水线并行（Pipeline Parallelism）用来拆分模型，数据并行（data parallel）用来拆分训练样本，一般来说，数据并行和流水线的通信量较小，一轮迭代在 GB 级别，模型的通信量较大，一轮迭代在上百 GB 级别。任务调度过程中不同节点和卡的分配对训练作业性能影响较大，如何选择最优的资源分配模型是对 AI 任务调度的巨大挑战。

资源碎片化：不同训练作业所需的资源不同，任务生命周期也各不相同，集群稳定运行一段时间后，不可避免出现较多资源碎片问题，导致在集群有空余资源的情况下，某些任务依旧无法运行。

（2）关键技术和价值

在 AI 训练和推理过程中，任务调度具有至关重要的作用，通过对任务进行合理调度，可以有效地提高计算资源的利用率，降低计算成本。云原生平台提供多种 AI 任务智能调度能力，通过组调度（Gang）能力避免不同训练作业之间资源死锁和忙等的问题；结合节点网络拓扑调度、Tor 亲和调度和超节点分组亲和调度能力，大幅度提升 AI 训练任务性能；装箱调度和重调度配合使用，缓解集群资源碎片过多的问题，提高整体资源分配率。

组调度（Gang）：组调度满足了调度过程中“All or nothing”的调度需求，避免 Pod 的任意调度导致集群资源的浪费。在 AI 训练场景中，如果某些训练的 Pod 没有被调度成功，已调度完成的 Pod 会继续空等，造成资源浪费、甚至资源死锁。Gang 调度会根据训练作业所需的最小资源量进行判断与调度，如果集群剩余资源不满足该作业所有 Pod 同时运行，该训练作业不被调度，直到集群资源满足该作业内所有 pod 资源需求后，作业才会被真正调度与执行。

节点网络拓扑感知调度：节点内卡与卡有多种通信方式，比如 GPU 卡之间的网络连接方式分别为

PCIe 和 Nvlink，其中 Nvlink 的通信效率数倍于 PCIe。训练作业内 Pod 申请多卡执行任务的场景，调度器应感知节点内卡与卡之间的网络拓扑类型，选择网络通信最优卡资源组合分配，保障训练任务性能最优。

超节点拓扑亲和调度：相对于跨超节点通信的场景，超节点内部的 GPU 卡之间具备高一个量级甚至更大的高带宽互联。随着大模型训练参数量级的快速增长，在模型并行度超过单节点卡数的场景，考虑将分布式并行策略与超节点拓扑结构相结合，模型并行任务控制在一个超节点内，超节点之间执行数据并行和流水线并行，分利用超节点内 GPU 卡的高速通信优势，提升大模型训练效率。

装箱调度（Binpack）：装箱调度主要用于解决集群资源碎片的问题，在 AI 训练场景下包含两种维度的装箱调度，分别为节点维度和 Tor 维度。对于不满 8 卡的训练作业优先进行节点级装箱调度，填满集群内节点资源碎片，空余更多节点用于调度需要整 8 卡的训练作业；对于单个节点无法承载的训练作业，优先进行 Tor 级别装箱调度，优先填满 Tor 级别的节点资源碎片，结合 Tor 亲和调度为后续训练作业分配整 Tor 资源，提升平台整体训练作业效率。

主动重调度：AI 训练和推理任务的生命周期各不相同，短则数分钟，长则数月。随着 AI 任务持续运行集群中不可避免出现节点资源碎片，在集群内资源碎片现象严重的场景下，大模型训练作业往往会因无法获取整节点资源而长时间排队。重调度策略主动识别集群碎片率过高的节点，驱逐并按照装箱策略重新调度该部分节点对应的任务，整合节点资源碎片。重调度涉及业务的驱逐与重启，对 AI 任务的故障恢复有一定的要求，比如具备 Checkpoint 能力，设置正确的 PDB（Pod Disruption Budget）策略等，确保业务在资源碎片重整的过程受影响程度最低。

2、AI 训练存储加速

（1）现状与问题

从过去的经典 AI，到现在的大模型，AI 模型的参数及 AI 算力规模呈现出指数级的爆发增长，对存储基础设施也带来全新的挑战。AI 训练业务上对存储的主要挑战如下：

大模型训练 AI 集群故障概率高，故障影响大，故障发生后任务恢复耗时长，浪费大量 AI 算力和训练时间。训练大模型涉及大量 GPU/NPU（成千上万个）和训练时间（以月为单位），因此作业失败在 GPU/NPU 训练集群中很常见，机器崩溃和网络故障不可避免。分布式深度神经网络训练作业被中断，将导致模型状态（即模型参数和优化器状态）丢失，训练作业失败导致模型需要重新训练，会浪费大量算力和时间。

训练任务检查点 Checkpoint 是深度学习框架中的容错方法。检查点 Checkpoint 通过在给定时间定

期保存完整模型状态的快照来帮助缓解训练的模型状态丢失问题。如果发生故障，可以使用之前保存的 Checkpoint 快照将模型重建到快照时的状态，以从该点恢复训练。

但 Checkpoint 检查点和故障终止之间的进度仍会丢失，因为必须重新执行计算以恢复未保存的中间结果。根据 Checkpoint 检查点保存频率，通常会导致几个小时的计算时间损失。此外，保存和恢复 Checkpoint 过程本身会产生大量开销，恢复时所有节点都需要读取 Checkpoint，千亿大模型 TB 大小的 Checkpoint 文件保存和恢复通常会成为训练过程中的瓶颈，Checkpoint 保存和恢复过程中会长时间中断训练任务，浪费大量算力和时间，考虑到大模型使用的 GPU 规模，以 1 万卡为例，故障损失将会是数万个卡的时间。

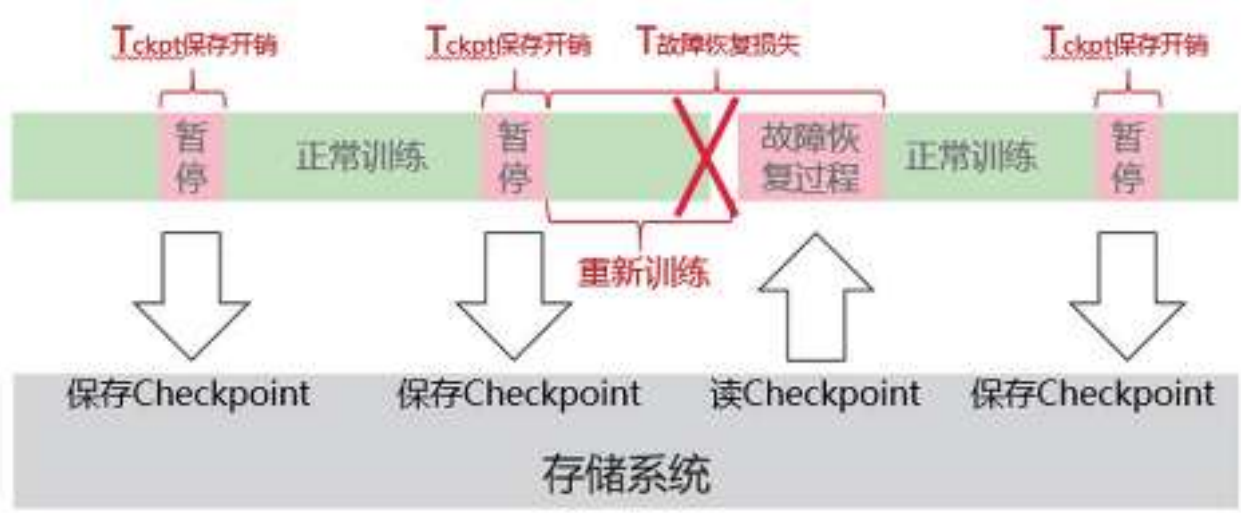


图 3-2-1　训练过程中 CheckPoint 保存和故障恢复时的时间和算力开销

更高的 CheckPoint 保存频率，更快的 CheckPoint 保存速度，以及训练任务故障时更快的 CheckPoint 恢复速度，可以有效节省 CheckPoint 保存及训练任务故障恢复时带来的算力和时间开销，加快大模型训练速度，提高大规模集群的算力利用率。

CV/ 多模态 / 自动驾驶等训练任务场景模型训练计算速度很快，但底层训练数据量很大，数据集读取慢导致 GPU/NPU 算力出现空闲，训练任务变慢。随着 AI 数据量不断增长，训练使用 GPU/NPU 越来越多，底层存储的 IO 已经跟不上计算能力，企业希望存储系统能提供高吞吐的数据访问能力，充分发挥 GPU/NPU 的计算性能，包括训练数据的读取，训练数据的读取要尽量读得快，减少上层算力对存储 I/O 的等待。

（2）关键技术和价值

针对 AI 训练场景中面临的挑战，我们可以提供基于大容量对象存储服务 + 高性能文件服务的 AI 云存储。

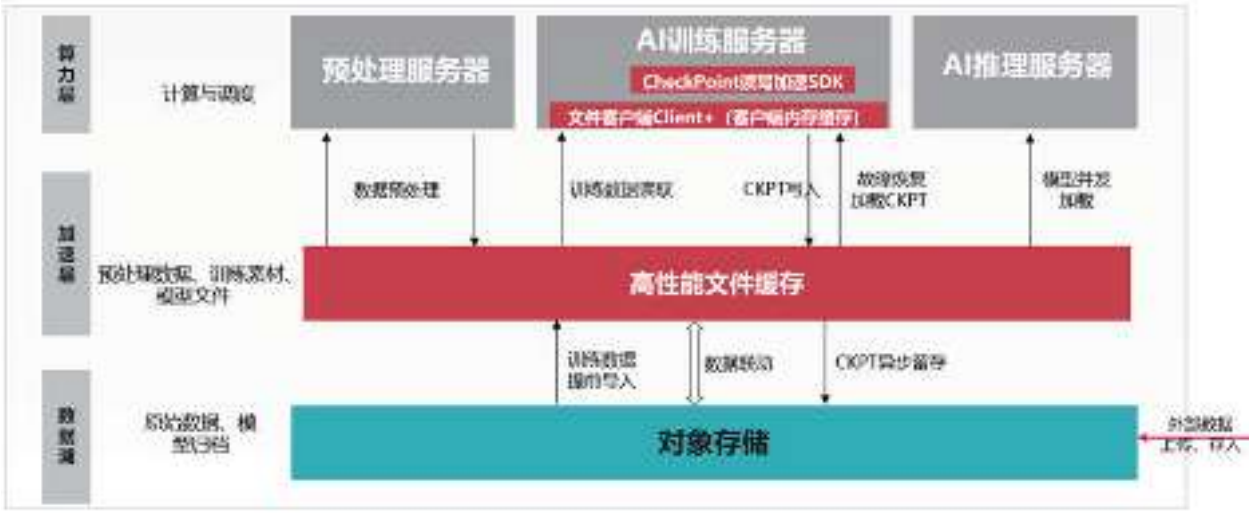


图 3-2-2　基于大容量对象存储服务 + 高性能文件服务的 AI 云存储架构图

首先，云原生时代，业务数据首先是集中积累到数据湖存储中，是数据入云的第一个落脚点，数据湖存储是 AI 数据采集和导入、大数据预处理、模型开发、训练、模型推理分发部署等各个业务环节进行数据流转的核心，能够方便地和各个业务环节进行数据交流，通过 S3/HDFS 多协议能访问到同一份数据，数据免搬迁，因此对象存储是数据湖存储的首选。

在数据湖之上，针对 AI 最为核心的训练环节涉及的海量小文件高性能、低时延，及大模型训练 Checkpoint 快存快恢诉求，提供一个高性能存储加速缓存层作为数据湖存储的补充，弥补数据湖存储在支撑 AI 高性能计算时性能不足的问题，加速训练过程。AI 训练在数据加载以及为了故障恢复做的 CheckPoint 机制中，AI 训练集群需要进行高性能数据读写，高性能文件缓存在 AI 训练场景是必须的服务，高性能数据缓存用于加载训练任务需要的数据集，保存 CheckPoint、模型文件等热数据，用于训练任务故障恢复，模型评测时基于近期 CheckPoint 进行训练算法微调等场景。

a）数据联动技术

高性能文件存储内置 Bucket Link 数据联动功能，高性能文件存储内的文件系统可以绑定容量层的对象桶，用户无需手工部署外部迁移工具即可实现在对象存储和高性能文件存储两个分布式存储服务之间进行高速数据流动，存储各节点均可参与数据导入、导出，数据流转比人工带外工具方式更加简洁高效。

AI 训练场景中，通过高性能文件系统来加速对象存储中的数据访问，用户通过指定高性能文件存储文件系统内的目录与对象存储桶进行关联，然后通过创建数据导入导出任务实现数据同步，训练开始前可以将对象存储数据湖中的 AI 训练集数据高速预热到高性能文件缓存加速层中，实现训练时数据集高速读取，避

免 AI 芯片因存储 I/O 等待产生空闲，提升 AI 芯片利用率。大模型训练过程中周期性产生的 CheckPoint 数据可以高速写入高性能文件缓存，减少对上层训练任务的中断和阻塞，可以提高 CheckPoint 保存频率，减少训练任务故障时需要从最近一次 CheckPoint 重新训练的损失，最后，高性能文件存储通过配置缓存数据淘汰功能，及时将长期未访问的数据从缓存中淘汰，释放高性能缓存空间。

数据联动技术包含以下功能：

元数据导入（Lazyload）	高性能文件系统绑定对象桶后，可以使用元数据导入功能。元数据导入功能仅会导入文件元数据，文件内容会在首次访问时从对象存储桶中加载并缓存在高性能文件存储中，后续重复访问会直接命中高性能文件存储缓存，无需再从对象存储桶中加载数据。元数据导入（Lazyload）方式下，初次访问时需要从对象存储中加载数据内容，对文件的第一次读取操作可能耗时较长，适合对首次数据访问时延不太敏感的业务场景。
数据预热（Preload）	高性能文件系统绑定对象桶后，也可以使用数据预热功能。数据预热（Preload）功能会同时导入元数据和数据内容到高性能文件存储中，上层业务访问数据时可以从高性能文件存储中直接命中，最大程度减少业务访问数据时的时延。如果上层业务对数据访问时延比较敏感，比如 AI 训练等场景涉及海量小文件，可以选择数据预热（Preload）方式。
数据导出	高性能文件系统绑定对象桶后，可以使用数据导出功能。当上层业务在数据联动目录里创建一些文件，需要将这些文件存储到对象桶里，可以使用数据导出功能。数据导出支持手工导出和配置成自动导出到对象桶。数据导出为异步方式，即上层业务将数据同步写入高性能文件存储，高性能文件存储以异步方式将数据导出到对象桶，不阻塞上层业务。
缓存数据淘汰	高性能文件系统绑定对象桶之后，可以配置数据淘汰功能，自动释放设定时间内没有访问过的文件数据内容，仅保留文件元数据，数据内容释放后不占用高性能文件存储的缓存空间，再次访问该文件时，将重新从对象存储中加载文件数据内容。数据缓存淘汰功能可以减少冷数据对高性能缓存空间的占用。

数据联动技术相比带外迁移工具的优势：

	数据联动	传统带外工具迁移方式， 如 cp、rsync 等
数据同步	快 系统所有节点参与数据同步、并发度高；不依赖外部中间商；	慢 依赖中转机 CPU/ 网络开销、客户端并发度；中转跳数多；
增量数据同步	极快 通过对象存储增量对象事件通知，只同步增量对象	极慢 需要全量比对找出增量文件、速度极慢
同步策略	可定制 比如只同步元数据、有些大文件 AI 场景下对象数据湖吞吐即可满足性能诉求、加速层只需要缓存元数据	不可定制、比如不能只导入元数据
调度器集成	简单可靠 同步发起、状态展示等通过服务控制 API 集成	困难、不可靠 中转机复杂环境下变数多、稳定性兼容性差、工具状态获取解析困难，调度器难以集成，人工维护成本高
垃圾回收	简单 通过调度器和系统可以自动释放资源	困难 同步失败会残留垃圾、需要手工清理、容易遗漏

表 3-2-3　数据联动技术相比带外迁移工具的优势

b）三级缓存加速技术

高性能文件缓存加速层存储提供 L1 服务端内存缓存，L2 客户端内存缓存，及专门针对 CheckPoint 快存快恢的 SDK，形成三级缓存加速技术，加速 AI 训练过程中的训练数据集读取，CheckPoint 快速保存及故障时的 CheckPoint 快速恢复。

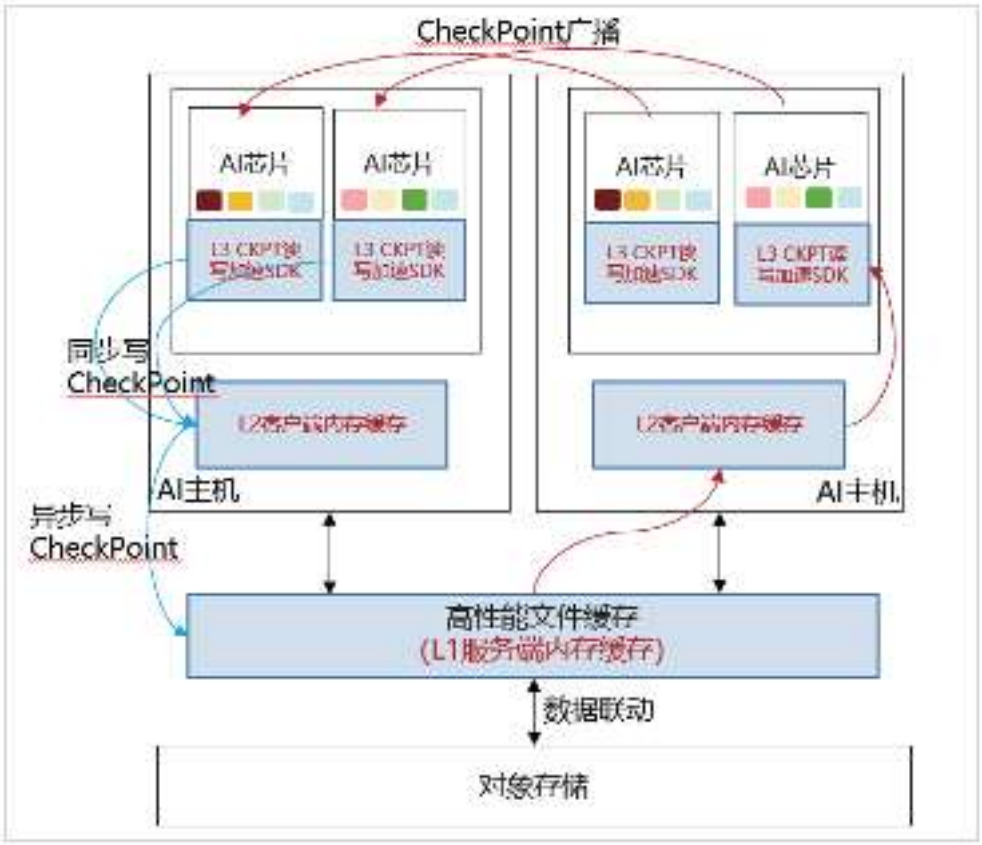


图 3-2-4　三级缓存加速技术

训练数据集访问加速：AI 训练过程中一般存在多轮 epoch，在每个 epoch 内，首先需要对数据集进行随机打散，然后将打散后的数据划分为若干 batch，每读取一个 batch 的数据，进行一次训练迭代，每轮 epoch 都会完整读取一遍数据集。因此多轮 epoch 之间，读取的都是重复的数据，只是访问顺序不同，而且 AI 训练数据集一般都是 KB 级海量小文件，数据总量较小，因此 AI 训练访问数据集时存在缓存友好特征。在业务访问数据集文件时，高性能文件存储会将 Nvme SSD 存储池中的数据文件缓存到 L1 服务端分布式内存缓存中，减小 AI 训练访问数据集的时延，同时在大规模训练集群并发访问数据集时，可以充分发挥 L1 服务端内存缓存带宽优势，实现比 Nvme SSD 硬盘层更大的吞吐能力。另外高性能文件存储的分布式元数据，可以支撑百亿级小文件扩展，也进一步缩短了海量小文件元数据操作的时延，提升了海量小文件操作的 IOPS 吞吐。

CheckPoint 保存及恢复加速：L3 CheckPoint 读写加速 SDK 组件针对进程级故障和 JOB 任务级故障等场景，对接 Pytorch/Mindspore/Deepspeed 等主流大语言模型训练框架，专门针对 AI 训练中的 CheckPoint 保存及恢复过程进行加速，实现 Checkpoint 先高速同步写到本机 L2 客户端内存缓存，再异步

持久化到服务端存储，最大程度减少 CheckPoint 同步保存耗时，减少了训练任务中断阻塞。AI 训练任务发生进程级故障时，利用本机 L2 客户端内存缓存实现 CheckPoint 原地秒级快恢，发生节点故障及 JOB 任务重调度场景下，利用客户端节点间高速参数面网络实现 CheckPoint 广播技术加速 CheckPoint 恢复速度，最大程度减少 CheckPoint 并发恢复耗时，避免训练任务故障恢复时由于远端存储带宽瓶颈导致长期阻塞。通过 L3 CheckPoint 读写加速 SDK 及 L2 客户端内存缓存技术，可以有效加速 CheckPoint 保存及恢复速度，可以提高 CheckPoint 保存频率，大大减少了故障恢复时需要从上一次 CheckPoint 重新训练的损失。

3、AI Serverless 训练

Serverless 作为一种云原生的应用开发和运行模式，能够将开发者从基础设施的维护中解放出来，专注于应用本身。这与 AI 算法工程师的需求契合，也能够解决 AI 训练推理任务面临的一些挑战。

（1）现状与问题

对于 AI 算法工程师而言，数据清洗、特征工程、训练调优是他们的强项，然而基础设施的准备和 AI 训练任务的部署，会给 AI 算法工程师带来额外的学习成本。

通用计算任务的算力需求比较容易评估，工程师可以根据业务模型提前进行性能测试，得出不同业务并发量梯度下的算力需求。AI 训练单次任务耗时长，模型收敛的速度和学习率、batch 大小等超参数相关，模型精度更需要通过多轮迭代优化才能得到理想效果。因此，AI 训练任务的算力需求不易提前获得。为了保证 AI 训练任务的效率，AI 算法工程师只能采取保守原则，预留足量的算力资源，往往造成算力支出的浪费。

通用算力场景常见的资源效率提升手段就是弹性扩缩，建立好单实例的业务并发量算力模型后，根据实际业务并发量进行实例数量的调整，避免业务低谷期空闲实例造成浪费。对于 AI 训练任务而言，在实际执行弹性扩缩时，存在并行任务分片重新调度的问题。训练集群弹性扩缩后，worker 的数量发生变化，取决于采用的并行策略，每个 worker 负责的数据、张量需要分配，数据分片，梯度状态等也需要重新同步。目前这一过程需要 AI 框架的支持和配合，任务重启和数据、状态的同步也会带来额外的训练开销。

（2）关键技术和价值

支持 Serverless 化的 AI 训练任务，除了常规的计算资源维度的调度，还需要引入拓扑感知的调度。这既包括节点、超节点内加速卡之间的通信拓扑，也包括跨节点的网络拓扑。AI 训练任务为了提升效率，都会采用分布式并行策略。每个 worker 负责一部分数据或张量的处理，各个 worker 完成后通过 gather/reduce 操作合并和同步各自的处理结果。因此，AI 训练任务各个 worker 的调度结果，应该和 AI 训练任务的分布式并行策略相匹配，避免 gather/reduce 阶段存在短板 worker 导致其他 worker 空等待，影响效率。

为准确评估 AI 训练任务的算力需求，为算力消费提供指导，需要引入 AI 训练任务的资源画像。以 AI 模型源代码为输入，提取模型的静态计算图。基于计算图进行算子拆分，对计算过程中的资源用量进行测算。同时根据设备网络拓扑和通信测量（包含带宽、时延等指标），构建网络通信模型。最后通过优化算法搜索满足 AI 训练任务 SLA 要求的最优资源配置和并行策略。从而简化 AI 算法工程师的模型部署流程，降低 AI 训练任务的成本，提升集群的资源利用率。

对于训练任务的弹性扩缩，业界有提出称作“透明弹性”的技术，以消减 worker 数量变化带来的开销。“透明弹性”，顾名思义就是弹性扩缩不改变 worker 的数量，而是通过加速卡虚拟化技术，调整 worker 共享的加速卡算力份额。举个例子，一个训练任务初始划分了 4 个 worker，每个 worker 各自独占一张加速卡，将其缩容为原来的 1/4，那其实可以将 4 个 worker 的执行都调度到一张加速卡上，分时复用加速卡的算力。当然，“透明弹性”的实现是需要一些关键技术的支撑的。首先，虚拟化层需要根据调度到的 worker 正确地换入对应的数据到显存；其次，显存的换入换出需要一些优化手段减少开销，比如识别各个 worker 共享的数据，这部分显存可以常驻；最后，worker 的调度还需要考虑并行策略下的依赖关系，避免死锁产生。

4、AI 故障自愈

（1）问题与挑战

大模型分布式训练任务周期以周或者月为周期，集群规模在千卡甚至万卡级别。在整个训练任务过程中会发生各种类型的故障，导致训练任务频繁中断，从而导致整体资源利用率低。总结下来，大模型训练在故障快恢领域面临如下挑战：

故障发生频繁：模型越来越大，大规模训练集群涉及的硬件数量达到百万至千万级别，特别是高网络带宽对光模块数量的要求越来越多，光模块的故障率相对较高，导致大量的硬件潜在失效风险。分布式训练是多个节点协同工作，任何一个硬件设备故障，都可能导致整个训练任务中断。

故障后恢复慢：大模型训练涉及到 XPU、网络、存储、软件等多种类型的软硬件故障。故障排查流程长，根因定位定界复杂。比如，Meta 训练 OPT 模型平均中断 12 小时[⑤]，理论训练需要 33 天。实际训练却使用了 90 天，期间出现了 112 次故障。其中主要问题是硬件故障，由于缺乏有效的故障快速恢复能力，训练人员只能不断地花费大量时间来重启训练任务，据统计手动重启约 35 次，自动重启约 70 次。

⑤ github.com/facebookresearch/metaseq/blob/main/projects/OPT/chronicles/OPT175B_Logbook.pdf

大量故障导致频繁 CKPT 回滚，算力利用率低：训练故障恢复过程中，关键在于模型状态的保存与恢复。传统的 checkpoint 方法由于保存耗时较长，往往导致训练效率较低。据相关分析称，OpenAI 在 GPT-4 的训练中使用了大约 2.15e25 的 FLOPS，在大约 25000 个 A100 GPU 上进行了 90 到 100 天的训练，其算力利用率约为 32% 至 36%[⑥]。这种算力利用率低的情况在业内更加普遍。

（2）关键技术和价值

故障自愈与训练任务快速恢复是一个系统性工程，涉及到准入检测、故障快速感知、任务快速恢复等多个阶段。

阶段一：准入检测

在训练任务开始之前或者新节点加入集群之前，提前识别潜在故障点，可以防患于未然。相应的技术包括：

性能压测：针对重点模块压测，检出并拦截慢节点或失效硬件。

基础环境检测：对节点的基础配置（如驱动版本、OS 版本等）、关键系统服务进行检测，提前识别不合规的节点并进行拦截。

阶段二：故障快速感知

AI 训练遇到的故障总的来说可以分为芯片故障、节点故障和带外故障。

芯片故障：结合不同 GPU/NPU 芯片的故障模式，Device Plugin 订阅芯片故障并主动上报到故障处理系统，是云原生通用的硬件故障感知系统，可以有效缩短芯片故障响应时间。

节点故障：除了芯片故障外，节点还会出现其他各种类型的故障，比如磁盘只读、K8S 核心组件异常等。通过 node-problem-detector（NPD）组件，可以全面的检测节点故障，对于检测的故障节点，可以通过污点的形式快速隔离并通过 k8s event 等形式上报故障。

⑥ github.com/facebookresearch/metaseq/blob/main/projects/OPT/chronicles/OPT175B_Logbook.pdf

带外故障：大规模 AI 分布式训练，一般需要高性能网络的支持。如果高性能网络出现故障，很难通过节点检测手段感知，需要构建交换机指标和日志采集与实时分析系统来检测带外故障。

阶段三：任务快速恢复

计算图和算子编译缓存：图编译缓存功能支持将图编译结果进行磁盘持久化，当训练任务重新运行时直接加载磁盘上缓存的编译结果，有效减少图编译时长，快速恢复业务作业。

在线复位快速恢复：部分卡故障支持热复位，比如 ECC-Error 故障。通过原地作业复位，无需任务重调度，可以提升任务的快速恢复能力，提升稳定性和连续性。

Pod/Job 重调度：对于无法恢复的芯片或硬件故障，需要将对应的训练 Pod 驱逐并重新调度到健康节点上。比如 volcano 调度器结合节点故障快速感知与隔离技术，可以实现训练作业快速重调度能力。

3.3 云原生 AI 推理系统建设要点

云原生 AI 推理能力体系涵盖了 AI Serverless 推理以及大型语言模型（LLM）的推理优化，旨在为用户提供高效、灵活且可靠的 AI 服务。然而，随着技术的深入发展和应用场景的复杂化，这些方面都面临着一系列问题和挑战，本节将具体介绍云原生 AI 推理方面的挑战并给出解决方案。

1、Serverless AI 推理

（1）现状与问题

AI 推理任务同样有采用弹性扩缩提升资源效率的述求。相比训练任务，推理任务单次任务耗时短，也是请求 / 响应的服务模式，类似通用计算的集群服务模式。问题在于，推理时长和资源消耗和用户的输入规模有关，仅以并发量建模业务的算力需求不够准确，还需要将推理时长的 SLA 考虑进来。另外 AI 推理任务运行前的模型文件加载是不小的开销，推理集群的弹性扩容还需要考虑冷启动时延的消减问题。

（2）关键技术和价值

AI 推理任务的单实例算力模型可以基于资源画像建立，而为了消减弹性扩缩时 AI 推理模型加载的冷启

动时延，需要引入基于容量预测的集群弹性扩缩，在业务高峰期到来前开始准备实例。集群容量预测可以采用基于统计的算法，基于集群历史的容量在一定的时间区间内计算如均值、峰值、方差等统计值，预测值由这些统计值组合计算得出。也可以采用传统的机器学习方法，如将排队论、傅里叶分析应用到历史数据的分析当中，提取出数据的分布规律和周期特征，以此进行预测。

消减冷启动时延的另一个方向是加速模型的加载。比如基于加速卡虚拟化的模型切换技术，对于单卡可以完整加载的小模型，在进行模型切换时，无需等待上一个模型的显存完全释放后，才开始下一个模型的加载。而是根据模型网络的分层结构将推理过程划分为传输阶段和执行阶段，不同模型的传输阶段和执行阶段可以在加速卡内流水线化并行处理，从而将模型的加载时延缩短至一个传输环节。又比如基于模型分层加载的预热机制，先测量不同层的加载时长，仅对加载时长较大的层进行预先加载，而非对模型进行全量预热，在推理任务效率和资源成本间求取一个平衡。

2、LLM 推理挑战和优化

（1）LLM 推理原理

LLM 大模型推理过程分为 Prefill 和 Decode 两个阶段。Prefill 阶段，会对 prompt 进行分词，KV Cache 矩阵计算，生成首 token；Decode 阶段，直接利用 Prefill 阶段计算出的 KV Cache 及首 token，自回归不断更新 KV Cache 以及生成下一个 token。随着 prompt 长度逐渐增大，算力密集型的 Prefill 计算时延将不断增加。

同时，模型越大，prompt 长度越长，不止权重参数占用的显存会增加，KV Cache 占用的显存容量也会不断增加，在长序列场景下，KV Cache 成为主要瓶颈，而 KV Cache 容量的增加又加剧了计算的访存时延。

转化为系统实现视角，参考云原生传统 AI 服务的架构，可以表示为（为了避免过多的细节，将负载均衡等一些常规的元素省略）：

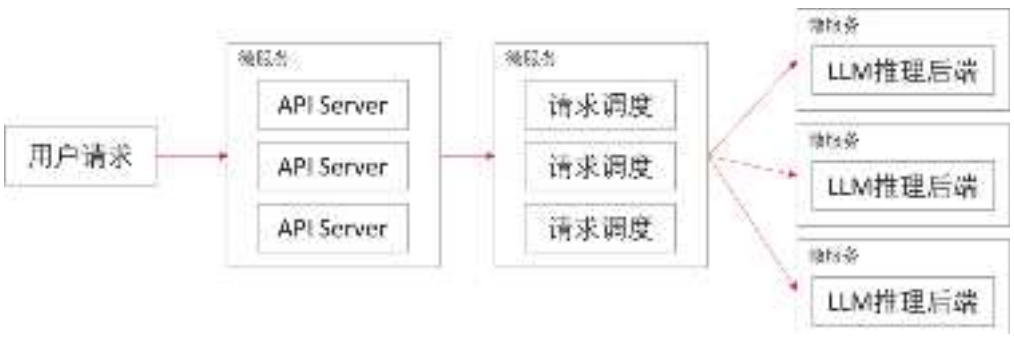


图 3-3-1　大模型推理系统基础架构

（2）现状与问题

AI 加速器内存容量增长速度远低于大模型存储需求的增长速度。典型的 Transformer 大模型的参数量以每两年翻 240 倍的速率增长，但是，业界典型的商用 AI 加速器的内存却只是每两年翻 2 倍[⑦]。大模型参数量和 AI 加速器内存容量增长速度之间的鸿沟，意味着训练和推理一个模型需要更多数量的 AI 加速器，这将大幅增加 AI 训练和推理的成本。另外，增加 AI 加速器的数量主要是为了让大模型能够在 AI 加速器内存中存储得下，这通常会造成 AI 算力的利用率低。

一方面大模型推理需要大量的显存，另一方面为了加速模型推理，还需要新的显存来做以存代算，比如 KV Cache 技术。

（4）关键技术和价值

显存扩展：通过弹性内存池扩展显存。推理过程中，将显存中 KV Cache、模型权重等 Tensor 动态卸载到大容量的共享弹性内存池。计算层与内存池通过高性能网络总线相连，同时将数据传输与计算过程流水线并行，有效降低了内存池访存开销。得益于大容量、高性能、共享访问的弹性内存池，显存扩展技术增大推理的批处理大小，从而提升 AI 推理的整体吞吐率。

计算卸载：卸载 KV Cache 相关的轻量算子到弹性内存存储服务，近内存计算机制显著减少了内存池与 AI 加速器间的数据迁移量。此外，基于 SLA 感知的卸载策略，智能决策卸载时机和粒度，通过弹性内存池大容量、大带宽的优势缓解了内存墙问题。

分布式推理： 张量并行（Tensor Parallelism）推理基于矩阵计算可拆分的数学原理，将推理计算切分成可以在多个专用加速器（DSA）上运行的任务，利用多卡的显存来解决单卡显存不足的问题。原理如图所示：

为了解决显存容量问题，LLM 推理后端需要从单体形态演变为分布式推理形态，以 TP 切分为 2 个分布式推理任务为例，如图所示：

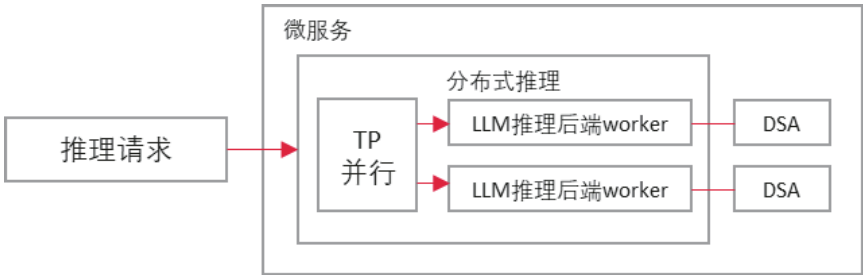


图 3--3-2　分布式推理后端

⑦ Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. "Ai and memory wall." IEEE Micro (2024).

而分布式推理系统，需要结合分布式计算框架 + 分布式推理两个方面的技术，前者将推理任务拆分为标准的分布式任务进行分发执行，并不关注具体推理逻辑。而分布式推理则着重于推理过程中，单个计算分拆为多个计算并行时需要考虑的问题，比如张量的切分，算子并行 + 集合通信等。

以存代算：随着系统提示词，多轮会话，长文本摘要等场景的普及，复用 KV Cache 成为一种必要的优化方案。例如将历史请求的 prompt 对应的 KV Cache 存储在弹性内存存储服务中，当具有相同前缀的 prompt 再次请求推理时，Prefill 阶段可以直接读取保存在内存存储中的 KV Cache，避免重复计算相同部分的 prompt，降低首 token 时延，提升端到端推理吞吐。

在输入比较多的场景，KV cache 将极大的节省计算时间。为了获取其收益，需要在系统中加入 KV Cache 管理技术：1）在计算之前，提前规划好需要申请的 KV Cache 地址空间，2）在计算的过程中，将根据输入计算得出的 K、V 值，存放在 KV Cache 中，3）下次需要使用对应的 K、V 时，从对应的 Cache 地址直接读取使用。

但这只是 KV Cache 管理的基本逻辑，放在分布式推理系统的上下文中，我们还需要考虑 KV Cache 在多个计算节点上的生命周期管理，甚至节点之间的 KV Cache 迁移问题。

因此从整个系统的角度，我们会需要一个新增的 KV Cache 管理微服务，来实现上述逻辑的管理，如图所示：

图 3--3-3　KV Cache 管理微服务

Prefill 和 Decoding 分离： Prefill 阶段，需要计算所有输入的注意力得分，大量的时间是花在各类矩阵乘法计算上，而 Decoding 阶段，由于 KV Cache 的存在，只需要计算新增的结果的注意力得分，计算量相对较小，如果 KV Cache 在整个系统中存在冷热分级的情况，Decoding 阶段更多面临的就是显存 IO 的压力。因此，在某些场景下，将大模型推理的组件进一步细分为 Prefill 组件和 Decoding 组件，依据计算侧重的不同，合理搭配计算和存储资源，进一步降低系统的资源消耗。这样推理后端又会产生变化，并且对 KV Cache 管理和调度都有新的配合需求。

首先要解决的就是 Prefill 和 Decoding 的计算如何衔接的问题。Prefill 的处理相对简单，跟不分离的推理过程一样，完成全量输入的首次推理计算，并生成 KV Cache。而 Decoding 阶段就变得比较复杂，首先 Decoding 阶段的输入，除了 Prefill 阶段的预测结果，其他全部存放在 KV Cache 中，而分离的场景，Prefill 和 Decoding 通常是放在不同的 DSA 上执行，此时就需要考虑 Decoding 如何访问 Prefill 阶段的 KV Cache 问题。最直接的方式，是将 Prefill 产生的 KV Cache 远程复制到 Decoding 所在的 DSA 的显存。

其次是 Prefill 和 Decoding 如何搭配的问题，因为 Decoding 的计算压力小，在某些场景，可能会存在一个 Prefill 搭配多个 Decoding 来发挥系统的最大吞吐量的情况。

3.4 云原生 AI 边缘云系统建设要点

1、边缘云原生技术与典型框架

边缘计算作为云计算的拓展，将边缘设备从数据消费者转变为兼顾数据生产者和消费者的双重角色，目的是满足能耗、隐私保护和实时性等方面的需求。由于软硬件分散部署在成百上千的不同位置上，管理这些分布式系统的方法课基于可观测性、松耦合系统、声明式 API 等范式，这些范式已经在云计算中展现出云原生技术的成功。

将云原生技术应用于边缘环境已成为主流趋势。作为云原生技术的底座，轻量灵活移植性高的容器技术天然匹配边缘资源受限且异构的场景，容器化的边缘部署模式将占主导。另外，自动化、自恢复的容器编排模式能够解决边缘资源分散而维护管理不便的问题。云原生丰富的技术生态亦可以轻松构建起监控、日志等能力，通过容器、流量控制和网络策略等能力能够有效提升边缘服务和边缘数据的安全性。

鉴于边缘环境的资源有限，将 Kubernetes 直接应用于边缘环境一般不可行。目前基于 Kubernetes 的边缘云原生技术平台的发展方向大致分为两种：其一是将 Kubernetes 进行轻量化，在边缘环境中实现与云

集群功能相似的边缘集群；其二是将边缘节点纳入云的集群管理，通过集成云边资源以部署系统。

将 Kubernetes 集群概念直接应用于边缘环境的一个轻量化解决方案是 rancher 公司开发的 K3s 项目。K3s 提供了对 x86_64、ARM、ARM64 等多种架构的支持，使得它可以运行在不同类型的硬件上，包括边缘设备和嵌入式系统。K3s 并没有对 K8s 进行大的修改及功能增强，因此 agent 组件与 server 间通信依然使用 list-watch 机制，这在边缘网络不稳定的情况下可能造成多次 list 的情况，造成比较大的通信负担。另外，K3s 并没有提供设备管理相关功能，这在以设备为中心的边缘环境中是一个巨大缺陷。除此之外，K3s 仅利用边缘资源形成集群，在某些场景下，边缘的资源可能不足以支撑应用的执行。

相比而言，云端资源可以作为边缘资源的扩充。因此，后一种云边协同的解决方案在业界较受欢迎，也孵化出许多的开源项目，其中的主流代表是 KubeEdge。KubeEdge 提供云端组件和边缘组件，通过两者之间的通信将云集群与边缘节点连接起来，形成一个整体的集群。在云端，KubeEdge 的组件旨在监听 Apiserver，管理边缘节点和设备信息，将集群的消息转化并下发至边缘节点。在边缘端，KubeEdge 对 kubelet 进行侵入式修改，提供了边缘轻量化方案以适配资源受限的边缘节点，同时提供了在边缘环境中非常必要的设备管理能力。

边缘计算的发展使得在靠近数据源的边缘设备上进行人工智能计算和处理（即边缘 AI）成为必然的发展趋势。边缘设备的资源受限，因此边缘 AI 目前多以云边协同 AI 的形式存在。为此，KubbeEdge 团队提供了云边协同的 AI 框架 Sedna。Senda 是一种结合云计算和边缘计算优势的人工智能解决方案，它通过协同管理和分布式计算将云端的强大计算能力与边缘设备的实时处理能力进行结合，以优化 AI 应用的性能、响应速度和资源利用效率。Sedna 构建在 KubeEdge 之上，利用 KubeEdge 的云边协同能力提供了云边联合推理、联邦学习、增量学习及终身学习等 AI 学习模式，并支持主流的 AI 学习框架。

2、面向边缘云原生的大模型轻量化技术

由于边缘设备的计算资源、存储容量和电池寿命受限，传统的大型 AI 模型无法直接部署和运行。因此，AI 模型压缩通过减少参数和计算量提升推理效率，能够适用于资源有限的边缘设备，有利于基于容器的模型部署与资源分配，以及模型的快速更新迭代。模型压缩方法如量化和知识蒸馏，可以在边缘设备上实现高效推理，满足实时性和资源限制需求。随着边缘计算和移动设备的普及，资源稀缺性问题如带宽限制、计算能力和能源消耗变得突出。高分辨率视频流和实时数据分析等资源密集型应用加剧了资源竞争，而强大的安全算法增加了通信延迟和能源消耗。边缘计算通过将任务卸载到附近节点缓解资源紧张，但动态拓扑结构和环境变化增加了通信性能和可靠性管理的难度。这些挑战要求在云边 AI 系统设计中采用灵活且适应性强的解决方案，以确保资源管理的高效性和稳定性。

为了应对以上挑战，业界提出了多种大模型轻量化技术，包括混合专家架构、边缘友好的量化、边缘感知的知识蒸馏和自适应模型压缩。

混合专家（MoE）架构在边缘环境中具有广泛应用，通过整合多种数据源和分布式专家处理，实现高效的数据处理和决策。该架构从各种输入数据（如 GPS、视频、生物识别、多媒体内容、激光雷达、速度信息、物联网数据等）中提取有用信息，并通过预处理和稀疏门控策略选择适当的专家进行处理。专家执行的任务包括但不限于自动驾驶、智慧城市、车路云、路径预测、环境监测。通过全局通信，所有专家共享信息和协调任务，系统根据性能反馈动态调整任务分配，确保高效运行。

边缘友好的量化通过减少模型权重或激活值的表示精度（从 32 位浮点数降低至 8 位或更低的整数），从而减少模型的存储需求和计算量。针对边缘环境的低功耗处理器，一般会采用 4 位或更低的精度，以提升计算效率和精度。

边缘感知的知识蒸馏通过训练一个较小的模型（学生模型）来模仿一个较大的模型（教师模型）的行为，从而获得一个轻量级但性能优异的模型。该方法也可通过分层知识蒸馏的方法逐层学习教师模型的特征表示，使学生模型在边缘设备上实现高效推理。

自适应模型压缩用于边缘设备工作环境多变、计算资源和电池电量也随时变化的场景。该方法可以根据当前设备的状态，通过启用或禁用某些神经网络层，或者调整量化精度，在性能和效率之间找到最佳平衡点。

3.5 弹性伸缩，应对 AI 任务浪涌挑战

在人工智能场景中，AI 任务的资源调度面临着许多不确定性，特别是在推理场景中，AI 任务的资源用量会随着业务需求动态变化，因此需要弹性伸缩能力以动态地分配和回收资源，节省成本。在业务高峰期，AI 任务可以通过扩容获得更多的实例以保证服务的正常运行；在业务低谷期，通过缩容减少 AI 任务的实例以节省成本。

AI 任务弹性伸缩包含的关键能力有资源画像、垂直弹性、水平弹性和智能 HPA 等。为了实现这些关键能力，需要对 AI 任务的资源消耗进行多维度、细粒度的监控，特别是在大模型场景中，除了对计算、内存、存储等资源消耗进行监控之外，还需要对 AI 任务的跨机跨卡的网络流量和带宽进行监控。然而，当前云厂商提供的监控工具（例如 Prometheus）无法提供对 AI 任务的算子级计算和通信进行细粒度监控的能力，因此还需要其他的监控工具（例如 Nvidia Nsight Systems）来监控 AI 任务的运行状态。

1、资源画像

（1）现状与问题

虽然容器技术提供了对 CPU、内存、存储等资源的抽象，容器编排系统为容器资源管理提供了声明式表达，但是为运行应用的容器选择合适的资源规格仍然是一个棘手的问题。在通用算力场景中，云厂商通过收集资源消耗的历史数据，通过统计学或者机器学习算法对应用进行资源画像，为容器的资源规格选择提供指导。然而，由于 AI 任务的复杂度更高，对 AI 任务进行资源画像面临新的挑战，特别是在大模型场景中，资源画像不得不考虑 AI 任务的并行策略。由于有部分模型跨机跨卡部署的特性，因此 AI 任务包含了不同类型的实例，且根据并行策略的不同，实例之间的相互依赖关系复杂多变。同时，由于实例之间存在依赖关系且需要频繁的数据同步操作，网络带宽在资源画像中显得至关重要。目前 AI 任务的资源画像面临以下的挑战：

动态负载	AI 任务的资源需求往往是动态变化的。在推理场景中，根据 AI 任务请求的变化，所需要的实例数也要相应地改变。此外，由于每个请求中需要处理的数据不同，单个实例所需要的资源也是变化的。如何捕捉动态负载的资源使用特征，需要在多场景、长时间的 AI 任务运行中收集数据以学习出更准确的资源画像。
模型结构	对于不同的 AI 任务，由于模型结构的复杂度不同，对资源的需求差异也很明显，因此需要资源画像算法能够对不同的模型进行学习。为了实现对模型结构的识别，要求资源画像算法能够对模型进行细粒度的切分和评估。

（2）关键技术和价值

多层次资源画像：在大模型场景中，当模型跨机跨卡部署的时候，AI 任务的资源画像不仅需要考虑单卡上计算子图或者算子的资源消耗情况，还要考虑跨机跨卡通信对资源画像的影响，因此需要一个将资源规格和并行策略都考虑在内的多层次资源画像，既可以从总体上对 AI 任务进行画像，也可以对分解之后的子任务进行画像，并将子任务画像之间的关系反映在总体画像之中。

预训练模型和迁移学习：为了让资源画像功能能够适用于更多的 AI 任务，可以借助预训练模型和迁移

学习等技术以减少画像算法适用于多种模型的学习成本。预训练模型通常仅仅需要少量的算力进行微调就可以完成特定的任务，可以高效地解决资源画像的普适性和泛化性的问题。

2、垂直弹性

（1）现状与问题

AI 任务的垂直弹性是指对单个实例的资源进行弹性伸缩以适应 AI 任务对资源需求变化，通常可以通过修改实例的配置文件来动态调整资源的规格以实现垂直弹性。为了实现 AI 任务的垂直弹性，目前面临以下的挑战：

有限的扩展性	一个实例可获得的资源规格存在上限和下限，约束了垂直弹性伸缩的范围。对于 AI 任务来说，单个主机上的硬件加速卡的数量是固定的，因此弹性扩展也受限于主机可用的资源量。
弹性伸缩的开销	对运行中的实例进行垂直弹性伸缩会引入不可避免的开销。由于对实例的资源规格进行调整时，需要中断当前实例并重启，因此需要对实例内的 AI 任务进行中断或者迁移。在大模型场景中，AI 任务的中断意味着需要回退模型的 checkpoint，会造成算力的浪费，在推理场景中，AI 任务的中断也会造成请求的重调度和 AI 任务的重新运行，增加了时延，降低了推理服务的质量。

（2）关键技术和价值

基于画像和预测的弹性伸缩：基于资源画像功能，对 AI 任务的资源需求进行预测，根据预测结果提前做好弹性伸缩的准备，实现更加准确和快速的垂直弹性。

混合弹性伸缩方法：将 AI 任务的垂直弹性伸缩和水平弹性伸缩结合起来，一方面对于可预测的负载需求变化可以使用垂直弹性伸缩方法；另一方面，对于垂直弹性无法满足需求的场景下，可以使用水平弹性伸缩方法，保证 AI 任务的正常运行。

3、水平弹性和智能 HPA

（1）现状与问题

AI 任务的水平弹性是指动态地增加或者减少实例的数量以适应 AI 任务的资源需求变化。智能 HPA（Smart Horizontal Pod Autoscaler）是指在水平弹性的基础上，在检测到度量指标（例如，CPU 使用率、内存使用率、硬件加速卡使用率等）发生变化时自动对 Pod 进行扩容或缩容。通过在资源管理平台中设置相应的触发条件，当 AI 任务的资源需求满足触发条件时，资源管理平台进行相应的动作以增加或者减少 AI 任务实例，从而实现水平弹性伸缩。虽然水平伸缩和智能 HPA 适用于大规模、需求变动大的 AI 任务，但是也面临以下的挑战：

网络开销

在大模型场景中，由于部分 AI 任务会跨机跨卡运行，不可避免地引入了网络开销，数据需要在不同的主机上的实例间传输和同步，因此进行水平弹性伸缩时就不得不考虑网络开销对整体 AI 任务性能的影响。

成本优化

水平弹性伸缩不仅需要考虑实例数对 AI 任务性能的影响，还要考虑实例数对成本的影响。实例数过少时，影响 AI 任务的性能表现；实例数过多时，增加了不必要的成本。因此需要综合考虑性能和成本来优化水平弹性和智能 HPA 的触发规则，在正常完成 AI 任务的前提下，减少成本。

指标选取和设定

实现 AI 任务的智能 HPA，不仅需要监控 CPU 使用率和内存使用率等指标，还需要监控 AI 任务相关的指标，例如硬件加速卡的算力和显存使用情况。此外，由于 AI 任务跨机跨卡运行，因此也需要监控网络通信指标。因此在设定触发指标的时候，需要考虑不同指标的相互关系和影响，寻找最优设定。

冷启动开销

由于扩展 AI 任务的 pod 的时候，实例需要一段时间来准备资源和加载 AI 模型参数，因此不可避免地引入了冷启动开销。此外，在 AI 任务中由于模型参数较大，加载时间长，因此冷启动的开销也较大。冷启动的时间会造成实例启动运行的滞后，从而影响到 AI 任务的性能表现。

（2）关键技术和价值

模型并行策略感知

在大模型场景中，模型的并行策略会影响不同实例间的通信情况，因此水平弹性伸缩增加或者减少实例的时候，需要感知 AI 任务当前的实例分布情况以及相互依赖关系，进而制定优化后的弹性伸缩计划。例如，对于 pipeline 并行策略来说，需要评估和识别出 pipeline 中哪一个 stage 是整个 AI 任务运行的瓶颈，从而对该 stage 的实例进行弹性扩展。

利用低成本实例

为了优化 AI 任务的成本，可使用低成本实例来进行弹性伸缩，例如 Spot 实例。由于这些类型的实例可以以折扣价格提供有效的资源，因此基于资源画像功能可以将执行时间较短的 AI 任务分配到这些实例上，在降低成本的同时也避免了这些实例因为资源抢占而对 AI 任务造成负面影响。

可容错机制

利用水平伸缩功能，在实例由于故障无法正常工作时，可以将故障实例隔离，同时创建新的实例来保障 AI 任务的正常运行，因此可以将水平弹性伸缩功能与故障检测和恢复功能结合起来，实现可容错机制。

智能扩缩容规则

基于资源画像功能，配合 AI 任务的性能表现监控，通过人工智能算法学习智能的扩缩容规则，部署于智能 HPA 中。同时，通过实时监控和增量学习不断改进智能 HPA，实现 AI 任务的自动扩缩容。

预热技术

为了减少冷启动对 AI 任务性能的影响，开发和使用预热技术，在资源需求低谷期保留一个最小数量的实例，而当预测到需求高峰即将到来的时候，提前创建和启动实例，加载 AI 模型数据，保障高峰期 AI 任务的性能。此外，为了减少模型镜像的大小和加载时间，可以对 AI 模型进行无损或者低损压缩，同时对容器镜像的大小也进行相应的优化。

PART 04

云原生 AI 技术典型应用场景

4.1 云原生 AI 跨地域多集群协同

42 云原生 A1 算力效能优化

4.3 云原生 A1 云边协同计算

4.4 大模型云原生化解决方案

4.5 云原生 A1 设备驱动管理

4.1 云原生 AI 跨地域多集群协同

1、应用场景

稀缺 AI 硬件资源未充分使用。GPU 等 AI 硬件资源分散在用户 IDC、不同厂商的公有云集群等不同位置，导致资源利用率不高。一般不同的集群静态分配给不同业务团队分开使用，当一个团队使用完后，资源处于空闲状态。而与此同时，其他需要使用资源的团队却无资源可用。即分布在多个集群的稀缺 AI 硬件资源不能灵活高效地被使用。

单集群可用性影响 AI 任务执行。虽然 Kubernetes 集群提供的节点、AZ 等反亲和部署能力，在单个节点或者整个 AZ 故障时，保证有一定的可用实例提供服务，客观上也提高了业务的可用性。但是当集群控制面和数据面组件故障时，特别是集群版本的原地升级时，集群内 GPU 驱动升级时，小概率的集群故障会引发了全局的业务断服宕机。这种故障会导致正在向用户提供服务的推理任务全部不可用，直接导致最终用户业务受损。

超大模型超出单集群资源上限。随着训练的模型越来越复杂，模型参数数量也越来越庞大，对于 AI 资源的需求越来越大，经常需要万卡以上的 GPU 集群。当资源规模超过一定量后这将超出单个集群的管理能力，如超出了单集群能管理节点上限。这就需要有一种机制在不能提升单集群管理规模的前提下，提供单集群兼容的的负载管理能力、AI 任务调度能力，满足大模型训练的需求场景。

2、解决方案

图 4-1-1　多集群架构

（1）分布式环境 AI 资源统一管理、全域调度，提高资源利用率

解决分布式环境资源分散、利用率不充分的问题的有效手段就是对这些资源的统一管理。即将分散的多个集群注册到一个舰队上进行管理，构建全局的资源视图，在舰队统一资源池上运行 AI 任务。基础设施管理无需分开维护一组 AI 集群，再静态地分配给不同的团队使用，而是将舰队分配个多个业务团队，在舰队入口上提交 AI 任务，舰队会根据调度策略动态地将任务分发到对应集群的对应节点上。

多集群的 AI 任务的调度支持分组调度，根据训练作业声明的最小资源在舰队管理的目标集群上选定资源，确保只有目标资源满足所有 Pod 同时运行时才会调度成功，否则不调度。同时调度算法也可以根据用户训练任务的管理特点进行装箱调度，优先填满一个节点、一个 TOR，或一个集群，从而减少资源碎片，提高全局的资源利用率。

此外，在多集群分布式环境下，通过描述各个集群的算力规模、资源类型、算力成本等因素，结合用户提交的任务的偏好，综合评价最优的源调度策略，通过最低的资源成本完成业务要求。如可以控制在任务调度时优先使用本地 IDC 内的资源，当 IDC 内资源不能满足要求时将任务分发到云上集群，从而实现充分的资源弹性目标，提高总体的资源利用率。同时，在多集群环境下，调度算法充分利用不同集群的不同资源特征，为训推一体提供了更多的调度选择灵活度。

（2）基于多集群构建高可用的 AI 作业运行平台

单个集群自身的控制面数据面故障发生的几率虽然比较小，但一旦发生将产生全局的业务故障。生产中的大部分案例发生在集群升级阶段，特别是集群数据面组件或驱动和现网环境特有的业务的兼容性问题在升级前测试阶段很难被发现。

基于多集群方案，两个集群组成一个舰队，用户在升级过程中，选择一个集群作为独立的灰度集群，通过舰队的流量和应用管理，将业务迁移到另外一个集群上。对灰度集群进行升级，确认在灰度集群上业务验证正常，升级成功后再升级另一个集群。这样在升级过程中即

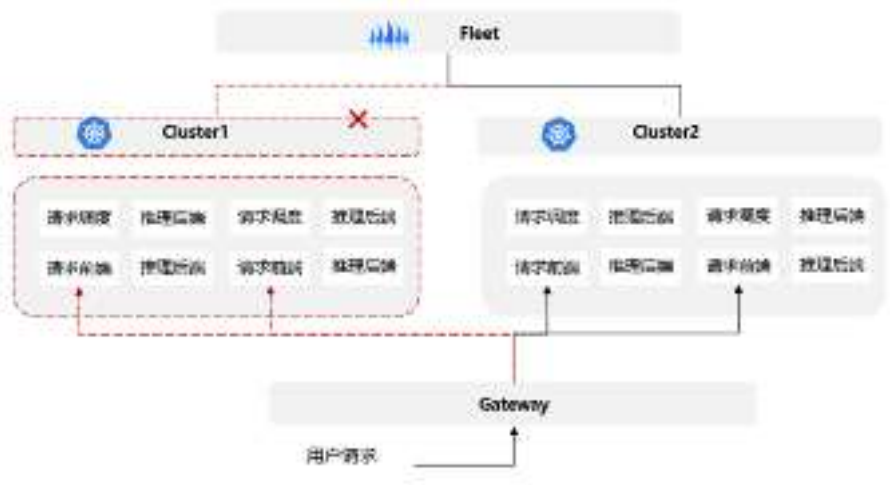


图 4-1-2　多集群功能架构

使出现问题，也不会导致用户最终业务不可用。

在 AI 推理场景中，基于舰队的多集群管理，当识别到一个集群全部或个别的推理服务实例异常，会自动执行故障倒换，将流量切换到另外一个集群上。同时将异常集群中的负载迁移到其他集群中，确保始终有足够数量的负载实例向用户提供服务，以确保服务的可用性。这种结合了流量迁移和负载迁移的方式，保障了用户推理业务的可用性，确保了总体服务质量符合用户期望。

（3）基于多集群构建大规模训练资源池

当单集群的资源上限难以突破时，解决当前超大规模和未来更大训练模型对资源极限需求的方案是通过构造基于容器舰队的多集群方案，提供资源的可扩展性。

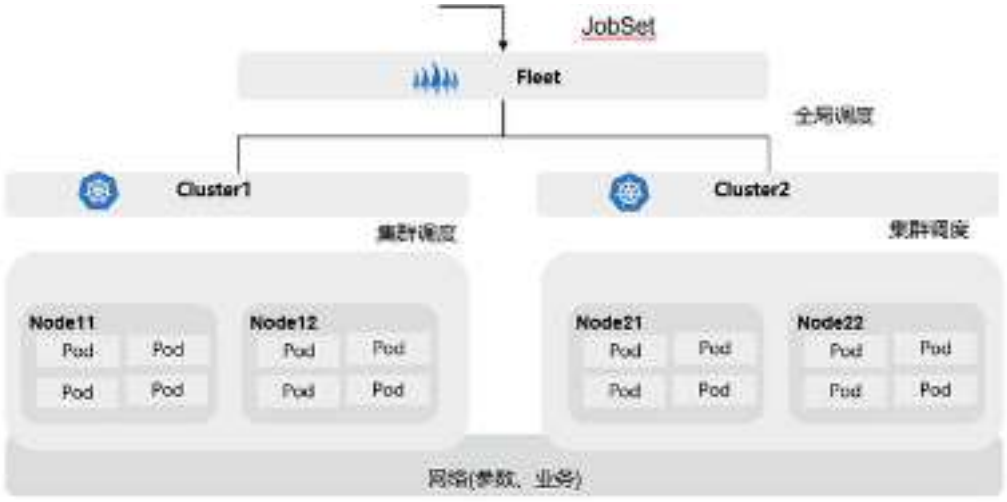


图 4-1-3　基于容器舰队的多集群架构

在舰队一层提供单和集群完全兼容的 API 和对象模型，像管理单集群一样在多集群上发放描述训练任务的作业集合和各种资源。在舰队层进行全局的作业调度，将作业集合拆分成作业分发到子集群。在子集群内基于集群的作业调度，将任务实例分发到对应的资源上。同时在全局资源视图上，当某个集群资源不足或故障时，基于舰队的全局调度可以在多集群间间协调，动态地在其他集群重新投放。当然在这种方式下，根据训练任务的需要，一般要求多集群资源的在一个参数面网络平面下，同时在同一个数据面网络平面下。通过这种方式可以突破单集群容量限制，大幅提升总体容量，满足大规模模型不断增长的需求。

4.2 云原生 AI 算力效能优化

1、AI 业务训推一体化

（1）应用场景

随着人工智能技术的不断发展，越来越多的企业开始将 AI 技术应用到自己的业务中，以提高效率和降低成本。在 AI 技术的应用中，推理和训练是两个不可或缺的环节，AI 推理业务直接面向客户交互，具有严格的 SLA 要求；AI 训练则 SLA 要求相对较低。

AI 推理业务存在明显的波峰波谷现象，用户请求频繁时业务处于波峰状态，资源需求量大，用户请求量减少，业务处于波谷状态，对应资源需求降低。为保证推理业务 SLA 要求，平台需根据业务峰值备货 NPU/GPU 资源，保障业务平稳运行，在次场景下，推理资源利用率极低，造成严重的资源浪费。

（2）解决方案

AI 训练和推理所需计算资源通过统一的集群管理，训推一体调度管理器通过资源分时复用与多业务驱逐策略统一协调计算资源运行训练和推理任务。

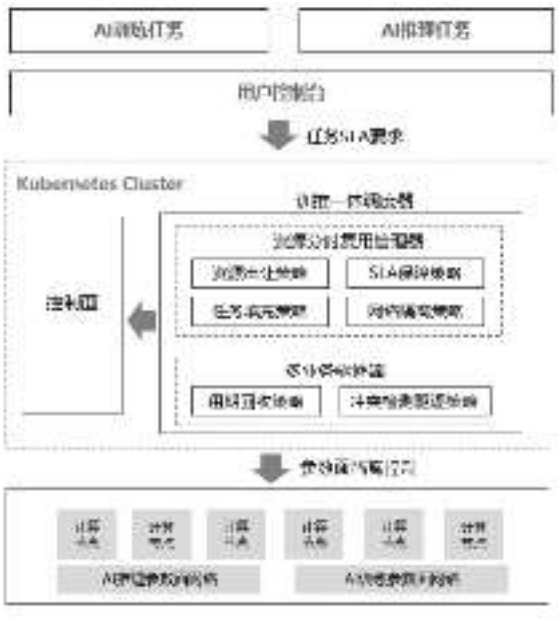


图 4-2-1　训练推理一体化架构

调度器结合以下策略与能力完成资源分时复用管理：

资源出让策略：调度器根据系统运行现状和用户自定义信息计算当前集群内可以共享给 AI 训练任务的资源信息，包括计算资源量和资源共享租期，用于填充 AI 训练任务。

系统运行现状包括但不限于 AI 推理任务的历史提交规律、实时负载信息等。用户自定义信息包括但不限于推理任务优先级、实时性要求、响应时延要求、任务标签分类等。

任务填充策略：调度器根据 AI 推理任务共享出的计算资源选择合适的 AI 训练任务进行资源填充，需满足两个要求，首先训练任务符合共享资源量要求并最大化使用该部分共享资源量，保证当前填充的训练任务拥有足够资源运行；其次训练任务的计划运行时长符合资源共享的时间段要求，并可以最大化利用该时段资源。同时对于满足上述要求的 AI 训练任务按照优选策略排序，选择排序最靠前的任务完成填充。优选策略包括但不限于训练任务优先级、实时性要求、响应时延要求、任务标签分类等。

SLA 保障策略：推理和训练具有不同等级的 SLA 要求，推理任务 SLA 要求较高，训练任务 SLA 相对较低，在推理和训练作业内也可以根据实际业务类型定义多种 SLA 等级，调度器结合业务 SLA 进行整体排序，保障高等级 SLA 任务优先获取集群资源，低等级 SLA 在资源不足时排队等待。SLA 参考维度包括但不限于时效性、可靠性、响应时间、修复时间等。

网络隔离策略：从安全性方面考虑，推理和训练混合部署运行需具备参数面网络隔离能力。当前参数面网络隔离方案通过节点的时分复用实现，调度器需分发业务时，需确保同一节点在同一时间点运行同一类业务，即单个计算节点内同时运行的业务是推理任务或者训练任务。AI 推理任务波谷时期，调度器共享部分资源用于填充 AI 训练任务，当 AI 推理波峰来临或者出现推理流量突增的场景，该部分资源通过多业务驱逐决策器快速回收给 AI 推理任务使用。

租期回收策略：多业务驱逐决策器根据 AI 推理任务共享资源的租期进行资源回收，到期后自动驱逐使用该部分资源的训练作业，提前准备资源接纳推理波峰期的任务请求。

冲突检测驱逐策略：多业务驱逐决策器进行业务冲突的实时监测，当集群内存在高 SLA 等级任务因资源不足而无法调度的场景，或者集群内运行中的业务 SLA 无法得到保障的场景，系统认定当前存在业务冲突，而后根据驱逐决策策略筛选合适的业务驱逐，用以保障整体业务 SLA。

多业务驱逐决策器根据以下三个方面选择被驱逐的业务：

a.SLA 保障

根据业务 SLA 由低到高排序，优先驱逐低 SLA 业务，在 SLA 等级相同的情况下，结合业务历史被中断次数进行排序，优先选择中断次数少的业务进行驱逐，避免同一个作业长期被中断而影响用户体验。

b. 业务代价评估

统计训练作业的运行时长，并根据同类业务的历史运行规律预估其剩余时长，优先驱逐开始训练时间短或剩余时长较长的训练作业。

c. 节点网络拓扑

结合计算节点网络拓扑（full mesh or partition mesh）信息，保障驱逐的业务尽可能分布在节点的网络高性能连通域内，释放的资源可以为新调度业务提供更高的计算能力。

2、算力资源共享：XPU 虚拟化

（1）应用场景

在现今单卡算力膨胀的时代，针对微量 AI 资源诉求场景，例如图像识别、语音识别等等，往往其资源诉求是无需占用一张完整算力卡的，那么如何让多个微量算力诉求的 AI 业务去共享 AI 算力资源，就成了一个必要课题。

多容器无法共享资源：

kubernetes 设备管理框架（Device Plugin）不支持多个容器共享同一个物理异构设备场景。k8s 原生 API 只支持整卡粒度为工作负载申请 XPU 资源，从资源管理层面上无法承载 XPU 虚拟化共享资源。

隔离性和安全性差：

NVIDIA MPS（NVIDIA Multi-Process Service）是 NVIDIA 公司为了进行 GPU 共享而推出的一套方案，由多个 CUDA 程序共享同一个 GPU context，从而达到多个 CUDA 程序共享 GPU 的目的。MPS 方案的致命缺陷，是把许多进程的 CUDA Context 合并成一个，从而导致了额外的故障传播。而且 MPS 共享无法实现资源隔离，一个占用大量资源的任务可能会导致其他任务等待时间过长，影响总体效率。

资源分配灵活性不足：

MIG（Multi-Instance GPU）是 NVIDIA 推出的一种 GPU 虚拟化技术，它允许将单一的 GPU 物理资源划分为多个独立的 GPU 实例，每个实例都可以有自己的 GPU 内存、计算单元和带宽。MIG 的一个限制在于它只支持固定比例的 GPU 资源切分。这意味着用户不能自由定义任意大小的 GPU 实例，而必须从预设的一组配置中选择。这可能不适合所有工作负载的需求，特别是那些需要非常特定或非标准资源配置的任务。

（2）解决方案

资源隔离：XPU 虚拟化技术的核心之一是资源隔离，它确保了在共享 XPU 资源的多容器或多任务环境中，每个任务或应用都能安全且独立地运行，不受其他任务的影响。XPU 虚拟化通过在内核层对 XPU 进行细粒度的切分，模拟出多个虚拟的 XPU 设备。这种切分不仅包括计算单元，还涉及到显存的划分，确保每个容器或虚拟环境都能获得专属的 XPU 资源份额，避免了资源争抢和潜在的数据泄露。XPU 与容器运行时集成，为每个容器提供了一个虚拟化的 XPU 环境。这意味着不同容器内的应用看到的是独立的 XPU 资源，如同它们各自拥有专用的 XPU 一样，实现了更高级别的隔离。

资源按需动态分配：XPU 虚拟化支持按需分配 XPU 的计算单元、显存等资源，可以将 XPU 资源切分成更小的单位进行分配，而不是仅仅以整个 XPU 或大块 XPU 资源为单位，提高了资源使用的灵活性和效率。XPU 虚拟化的动态分配是指在 XPU 虚拟化技术中，能够根据运行时应用的实际需求，实时调整和分配 XPU 资源的能力。这种机制使得资源能够更加高效、灵活地被多个容器或虚拟机共享，同时确保每个应用都能获得足够的计算资源来完成任务。

XPU 共享资源监控：对于容器化 XPU 共享应用，要能针对容器粒度和 XPU 分片粒度获取 XPU 资源使用情况，其中关键监控指标包括 XPU 算力使用率、显存使用量等。XPU 虚拟化监控指标与 AI 云原生基础设施监控系统集成，以便于统一管理和可视化展示。

3、算力资源共享：XPU 显存池化

（1）应用场景

AI 云原生场景下，AI 业务的 AI 资源诉求颗粒度，往往不是整数卡，这就导致了显存在使用过程中，存在和通用计算资源一致的问题，存在碎片现象、频繁申请释放开销现象。

显存碎片：显存和通用物理内存一样，在物理上存在连续显存的概念，当应用程序使用显存时，不同的应用程序规律不同，因使用显存的大小、申请释放的频率不同，不可避免的导致产生显存碎片。而不能跨 XPU 卡共享显存会导致显存问题加剧。

申请释放开销：显存的申请释放涉及到从软件到硬件的开销，包括软件申请时不同特权级切换的 CPU 开销，内核态访问物理资源带来的 PCI 带宽开销，以及显存申请释放过程中需要清零和刷新显存操作带来的开销。因此和解决通用内存的思路相似，需要尽可能减少应用在使用显存时涉及的各个节点影响，最通用的思路是建立不同程度的显存池。

显存故障:显存和通用内存一样，不可避免的存在故障。常见的场景有，部分显存区域可能存在故障、整块显存区域故障、访问显存的通道故障。需要提供技术能够更高效地发现显存亚健康状态，通过提供故障亚健康检测机制，故障快照和恢复机制减少故障带来的影响。

（2）解决方案

应用程序显存池化：应用程序对显存的管理，可以根据应用程序是否修改适配分为两种。

1. 应用程序侵入式修改，该种方法下，不同的应用程序均需要修改代码，实现显存管理的机制，对绝大部分应用程序来说会有极大的开发成本。

2. 应用程序显存池化的方案，不需要修改应用程序自身，基于应用程序和 XPU 基础库的运行机制（用户态 API 调用方式），增加显存管理的特殊处理流程，可以实现 API 使用维度的显存池化，即可以本地使用（增加可靠性加固）、也可以通过高速网络协议访问其它物理节点的显存。用户态 API 池化方案，可以实现 XPU 显卡在单物理机场景，多物理机场景的显存池化。

设备层显存池化	如应用程序显存池化原理介绍的所述，设备层显存池化属于驱动侧对显存的管理，设备层显存池化原理基于驱动的实现框架，内存管理机制在设备层做内存 API 的处理，不仅可以在单个物理机上将其它 XPU 的显存统一管理，统一映射池化，还可以借助高速互联协议，将其它物理节点上的 XPU 卡显存统一池化管理。
显存池化高可靠性	和传统内存面临的问题一样，显存在使用中显存硬件故障的检测，硬件故障的容错需要经过可靠性加固处理，以减少显存异常对业务的影响。可以通过两个关键技术实现可靠性增强。 a）通过提供显存压测工具，通过不同频度的压测来检查显存的亚健康程度。 b）通过定期支持显存快照，在显存异常后，异常上报系统支持应用做业务备份恢复，最大程度减少显存故障影响。

4.3 云原生 AI 云边协同计算

前文已经指出了边缘计算的发展方向，本章将结合具体的应用场景给出解决方案。

1、应用场景

边缘 AI 的应用场景可覆盖工业互联网、智慧交通、智慧城市以及 AR/VR 等各个领域，相较于云数据中心，边缘应用场景要更复杂。

在工业场景中，AI 可以应用于生产线设备的故障预测、产品质量检测等，通过实时数据分析与推理，提高生产效率与质量。然而工业设备通常存在数据无法共享、网络瓶颈等问题，导致数据集在地理上天然分割，传统的集中式 AI 模式无法共同使用多个设备的数据。同时单个设备样本数较少，普遍会出现冷启动问题。

在智慧交通场景中，AI 可应用到智能驾驶的车载摄像头、传感器等，辅助实现目标检测、轨迹检测等，但这些场景对实时性要求较高，传统的 AI 技术需要将采样数据上传到云中心带来的高时延无法满足毫秒级的

实时响应需求。

另外对于 AR、VR、互动直播、视频监控等场景，数据量大，资源用量大，对安全隐私的要求比较高，无法实现将所有采集数据上传到云端。

基于以上应用场景，面对边缘 AI 可能存在的资源受限、数据孤岛、小样本以及数据异构等问题，主要通过云边协同 AI 来解决在边缘训练的精度、时延、数据隐私等问题，在资源受限条件下提升模型推理性能。

2、解决方案

云边协同 AI 基于云侧算力和边缘侧数据合作完成持续推理和训练，以云端知识库作为中心，实现跨边的知识共享并处理边缘端任务，同时持久化和维护云端知识。

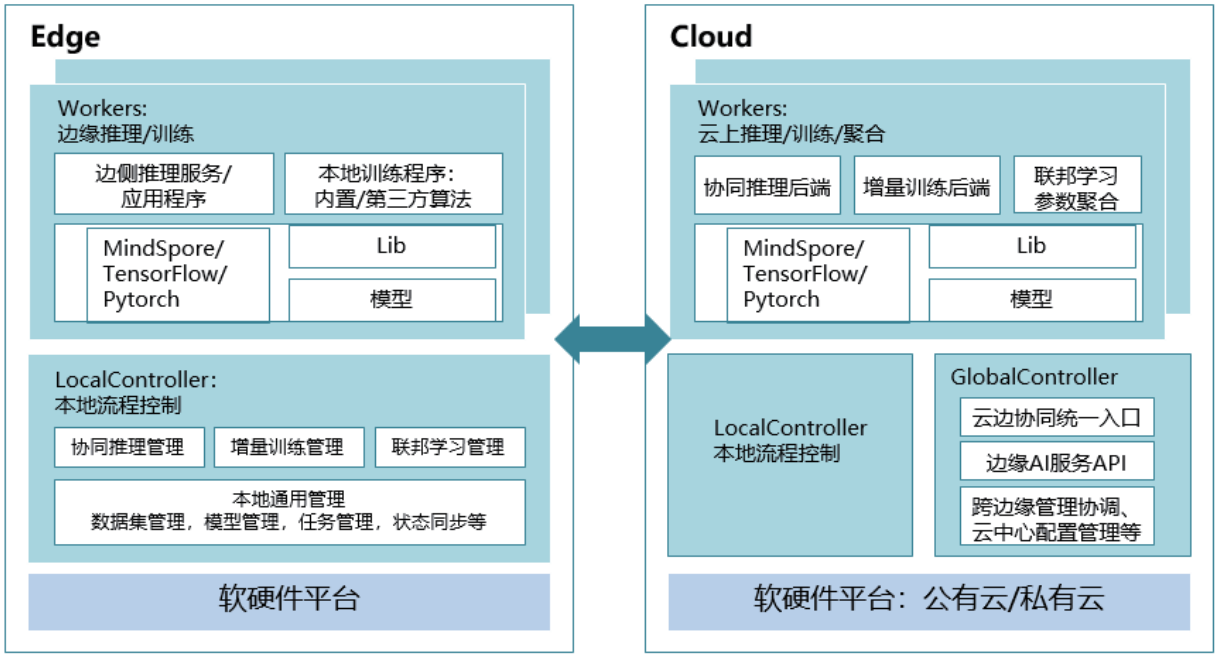


图 4-3-1　云边协同 AI 框架

云边协同 AI 框架从接口、性能、算法等多个方面进行优化。在接口方面，框架提供云边协同的 lib 库代码，兼容主流框架包括 TensorFlow，pytorch 以及 MindSpore 等；在性能方面，针对云边协同场景进行了云边通信、边缘存储的优化，使得云边协同更高效；在算法方面，集成了增量学习、联邦学习等多种适合边缘的训练推理算法，以下从协同推理和协同训练两个方面介绍云边协同 AI 的关键技术。

（1）云边协同推理

协同推理将推理性能较强的云端作为推理后端来提升推理精度。对推理来说，边缘端的推理可以获得更短的时延和更高的吞吐量，而云上的推理则可以获得更好的推理模型。云边协同推理通过难例检测算法对边缘端新数据进行判别，将简单样本在边缘端推理，难例样本发送到云端处理。简单的样本通常占大多数，保证了推理的时延和吞吐；难例样本占少量但容易出错，通过云上的大模型来推理出更准确地结果，使得整体精度得到提升。

（2）云边协同训练

a）云边协同的联邦学习

基于云边协同的联邦学习通过 GlobalController 跨多个租户从空间的维度帮助提升模型效果，原始数据在边缘端进行本地推理，再将模型参数传到云端进行汇聚，以充分利用分散在不同边缘端的数据。主要用于解决 AI 算法在工业落地时所面临的“数据孤岛”问题，在保障大数据交换时的信息安全、保护终端数据和个人数据隐私的前提下，在多参与方或多计算节点之间开展高效率的机器学习，实现各联邦学习参与方的联邦建模与利益共享。

b）云边协同的增量学习

针对边缘小样本和边缘数据异构的问题，云边协同的增量学习对单个租户从时间的维度帮助提升模型效果。通过增量学习，云端持续在边缘端监控生成的新数据并学习新知识，利用云边协同框架对边缘端推理模型进行更新，不断提高模型训练精度，同时缩短边缘端模型更新时间。

通过云边协同 AI，无需直接将所有边缘端的原始数据全部传输到云端，也可避免将原始数据从其本地边缘端存储库传输到其他边缘端系统中执行推理，对于由于数据隐私或者合规等原因导致数据无法迁移时尤其关键；可利用联邦学习或者迁移学习，基于所有边缘端的完整数据集进行高精度建模；可基于历史和更新数据，持续训练和改进边缘端的机器学习模型，在边缘端自动补充数据实现闭环。

4.4 大模型云原生化解决方案

1、应用场景

在 LLM 之前，流行的神经网络模型如 ResNet 50 的参数规模在 2 千万左右[⑧]，BERT 的参数规模在 3 亿左右[⑨]。而 LLM 大模型参数规模已经从十亿、百亿增长到万亿规模[⑩]。模型的大小也从 MB 增长到 GB 乃至 TB。

模型如何快速部署，快速提供推理 API，也成为一个 AI 服务提供商的棘手问题。大模型对并行计算硬件的需求已经不再是一块小的推理芯片能解决的，甚至需要多块高性能 GPU 和 NPU 协作，才能完成大模型的推理任务。

以容器和 kubernetes 为核心的云原生技术，因其在模型打包封装、实例弹性扩缩容等方面的优势，已经成为 AI 应用部署的首选。本章将介绍在云原生环境中，如何实现大模型的快速部署，并提供性能参数监控和运维能力。

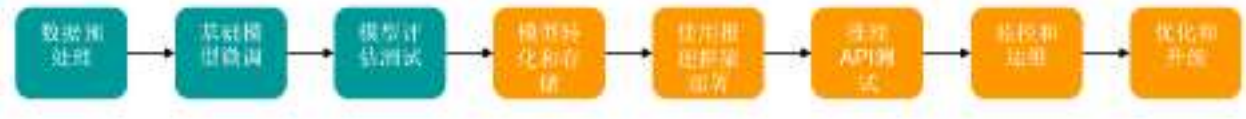


图 4--4-1　大模型工作流程

AI 大模型从训练到部署。用户在对基础大模型进行微调和优化以后，需要将其转化为可部署格式，并存储到响应网络存储位置 (不推荐直接打包到容器镜像中)，根据部署框架的不同，模型的参数存储格式可能有差异。随后根据不同的部署框架，准备推理的镜像，并选择环境进行部署。部署后通过 ELB 等方式暴露推理 API，并进行功能、性能和安全性等方面的测试。通过测试后可以上线到生产环境，并通过云原生环境提供的各种监控和运维手段，实时监控推理集群的资源占用和性能瓶颈。配置相应的弹性策略实现推理实例的自动弹性扩缩容。最后，通过包括模型推理的准确性、吞吐和时延性能等数据，优化模型并重复上述步骤在部署环境中升级模型。

⑧ https://arxiv.org/abs/1512.03385

⑨ https://arxiv.org/abs/1810.04805

⑩ https://huggingface.co/meta-llama/Meta-Llama-3-70B

2、解决方案

（1）推理引擎和部署框架

在 LLM 时代之前，已经出现了很多推理引擎如 Tensorflow Serving、TorchServe、Triton 等可以实现 AI 模型的部署。但是随着 LLM 大模型的退出，新的推理引擎如 vLLM、TGI(Text Generation Inference)、RayServe 因其对大模型推理的优化、更好地适应大模型接口和数据格式、以及大模型的分部署推理诉求等，获得了更多的关注。而这其中，vLLM 又因其极高的吞吐和高效的显存使用率，以及和 HuggingFace 的无缝集成等特性，成为当前大模型推理部署的首选方案。

云原生环境下，如何快速将如 vLLM 这类引擎部署到 kubernetes 集群中？ KServe 是构建在 Kubernetes 之上的模型推理部署平台，KServe 提供了一系列自定义资源 (CRD)，用于管理和提供机器学习模型的服务。通过使用标准化数据平面协议，为 Tensorflow、XGBoost、ScikitLearn、PyTorch、Huggingface Transformer/LLM 模型提供高抽象接口来部署生产级模型服务。它构建在 Istio 和 KNative 之上，屏蔽了应用扩缩容、网络、运行状况检查和服务器配置的复杂性，为大模型部署带来 GPU 自动扩缩容、和 Scale to Zero、灰度发布等尖端服务功能。当前它已经集成 vLLM 和 TorchServe，实现大模型的云原生部署。

（2）模型文件的存储和加载

模型通常是一些大文件，随着模型参数的增加，模型文件的大小也从几十 GB 增长到 TB 级别，通常采用 pickle 或 safetensor 等格式存储。当采用云原生环境部署时，需要将模型存储到云上，当前通用做法是将模型存储到成本相对较低的对象存储系统中。虽然对象存储价格低廉，扩展性好，但是对于模型的加载却并不友好。

模型加载就是将模型完整读取载入显存的过程，在推理服务部署时需要模型加载过程。加载过程需要从存储系统中单线程顺序读取，影响速度的关键因素是单线程顺序读取时的吞吐量。而对象存储由于远离运行中的实例，导致其加载速度受到网络带宽等因素限制。尤其是在 Serverless 场景下，推理实例无法快速启动，将影响集群响应推理 API 的时延和成功率。

因此对于存储在对象桶中的大模型文件，如果要实现 Serverless 部署，还需要在近计算侧部署一套缓存系统，如开源的 JuiceFS、AWS 的 Amazon File Cache、Google 的 Cloud Storage FUSE、华为云的 SFS Turbo 等应用。在近计算侧缓存模型文件，实现快速加载的目标。

（3）推理集群的监控和运维

云原生技术为应用的监控和运维提供了丰富的组件支持。对于推理性能的监控，可以通过 Prometheus+Grafana 实现可视化的性能监控 dashboard 和丰富的组合查询能力。对于分散在推理集群的大量节点的日志，可以采用 Elastic+Logstash+Kibana 软件栈实现自动收集、展示和查询。KServe 基于 Isitio 和 KNative 实现流量治理、灰度发布等能力，对推理集群的版本管理，流量控制，负载均衡等能力实现精细 的控制。

4.5 云原生 AI 设备驱动管理

随着 AI 应用的爆炸性增长，XPU 已成为加速深度学习、机器学习和其他高性能计算任务的首选硬件。GPU 驱动程序是 GPU 硬件与操作系统及上层应用之间的桥梁，它负责管理和优化 GPU 的资源分配、指令执行、数据传输等工作。对于 AI 应用而言，这直接关系到深度学习模型的训练和推理速度。GPU 驱动程序所处位置如下图所示：

1、应用场景

有效的 GPU 驱动管理对于确保 AI 系统的稳定运行、最大化计算效率和促进创新至关重要。GPU 硬件厂商会定期更新驱动程序来修复已知的问题和安全漏洞，引入新的特性，优化硬件性能，因此定期更新 GPU 驱动程序对 AI 系统非常必要。通过持续优化和升级 GPU 驱动，可以最大化地发挥 GPU 在 AI 计算中的潜力，支撑 AI 技术的快速发展和广泛应用。AI 基础设施应该具备方便地进行驱动程序版本管理和更新能力，以确保 AI 基础设施的长期安全可靠。

兼容性问题：云原生基础设施管理的资源通常比较复杂，包括不同的 GPU 硬件，不同的操作系统版本，不同的内核版本和不同的应用场景，以上都是影响 GPU 驱动兼容性的因素。这样就导致云原生基础设施中存在各种各样的 GPU 驱动程序版本，对运维人员造成来说是个非常大的负担。

自动化管理缺失：AI 大模型训练场景需要成千上万的训练服务器 ，GPU 驱动程序以服务器粒度进行安装和升级。当前云原生基础设施缺少 GPU 驱动程序管理自动化管理流程，驱动程序安装，更新成本非常高。

驱动升级影响业务：GPU 硬件驱动升级通常都会导致服务器上的 AI 应用中断，也就是 GPU 驱动升级是个冷升级过程。在 AI 推理和训练场景下如何减小 GPU 驱动升级对上层业务的影响，实现 GPU 驱动平滑升级，成为了 AI 基础设施面临的一个重要问题。

2、解决方案

在 AI 云原生时代，为了解决以上问题和挑战，GPU 驱动自动化管理是大势所趋，相关关键技术有如下几个：

（1）统一兼容性管理

AI 大模型正处于高速发展时期，云原生基础设施管理的异构计算硬件和软件环境的多样性日益复杂，为了避免因驱动兼容性问题导致 AI 业务失败，需要统一管理驱动程序的兼容性依赖如硬件型号，操作系统，内核版本等，建立驱动程序和兼容性依赖的匹配关系，运维人员可轻松获取 AI 业务可用的驱动版本和推荐的驱动优选版本。

（2）容器化驱动管理

利用 kubernetes 的应用编排能力，将驱动安装和升级流程容器化实现驱动管理自动化。kubernetes CRD 和 Operator 架构可以完成驱动管理任务的分发，实现驱动程序完整的生命周期管理，实时同步驱动程序状态，有助于自动化，可视化管理 AI 系统驱动程序。

（3）升级驱动联动应用编排

基于 K8S 容器编排平台的驱动管理系统可联动云原生节点管理和应用管理能力，实现 AI 业务无感的驱动程序升级。对于无状态应用比如 AI 推理业务，升级驱动前将 AI 业务驱逐到其他节点，分批滚动升级节点，可将业务影响降到最低。此外对于不可中断的业务可制定闲时升级策略，节点上 AI 业务全部结束后进行驱动升级。

PART 05

云原生 AI 行业实践

5.1 社交平台 RB 云原生 AI 平台应用加速实践

5.2 AI 解决方案提供商 FP 多场景 AI 云原生化实践

5.3 医疗科技公司 HL 云原生 AI 智能医疗实践

5.1 社交平台 RB 云原生 AI 平台应用加速实践

1、业务场景

社交平台 RB 基本上涵盖了互联网行业的所有典型场景，包括大规模搜索业务、推荐业务、广告文案、笔记生成、内容审核、封面生成、图片 OCR、问答机器人、智能助手等，该公司针对不同的业务场景，结合最新的大模型技术以及自身的平台，构建了云原生 AI 平台。

2、技术架构

云原生 AI 平台分层架构如图 5-1 所示，平台层包含了 AI 模型从开发、样本生成、模型训练、数据管理、模型存储、数据管理、配额管理、推理服务等端到端的功能，以及在训练推理过程中的故障快速发现、告警功能。平台层根据业务不同，提供了深度学习平台和搜索、推荐、广告平台。平台层帮助算法工程师高效的实现模型开发、训练、快速部署上线。平台同时稠密模型和稀疏模型的训练和推理。在资产集群层，RB 针对训练、推理服务对性能、SLO 的不同诉求以及异构硬件的特征，提供专有集群。在调度层，RB 进行了深度的优化的建设以支持跨平台调度如稀疏、稠密平台，支持在线、离线业务统一调度，并且提供配额、优先级调度能力以及失败重试功能。工具提供灰度发布、命令行以及智能的巡检。

图 5-1-1　业务架构图

Kubernetes 提供了良好的异构资源统一管理的能力，RB 在此基础上管理了 CPU、Nvidia GPU 等多种异构资源。在资源层，RB 采取了典型的多云方案，提供了丰富的存储类型以及跨机房、跨云、跨地域的多种网络传输类型。

随着 RB 业务的快速增长以及大模型技术的引入，RBAI 平台面临的两大核心的挑战：一是，如何综合考虑训练、推理平台，提升整体异构资源的利用率，降低成本？二是，大规模分布式训练中，如何在单任务达到千卡甚至万卡规模时，保障高计算效率和高稳定性？

3、实践一：统一调度、多业务混部、VGPU

图 5-1-2　调度业务架构图

调度器实现了对 AI 任务和在线任务的统一调度，然后将训练任务和近现任务部署到一个集群进行混部，在近现业务的低谷期，调度训练任务填充到闲置的资源，保证 GPU 资源的高利用率。在线服务或近现服务对时延敏感，对 SLO 有更高的要求，如果在提升利用率的同时保障在线服务或近现服务的 SLO 是一个关键的问题，混部平台提供了热点检测、快速避让，离线 GPU 压制等技术，在线任务业务洪峰来临，混部系统及时识别干扰和资源竞争，将离线任务进行压制，极端情况下，会根据算力优先级，驱逐优先级低的离线任务保障在线服务的快速响应，该方案在该社交平台取得了显著的收益，如图所示，整体的集群利用率保持在 80% 左右。

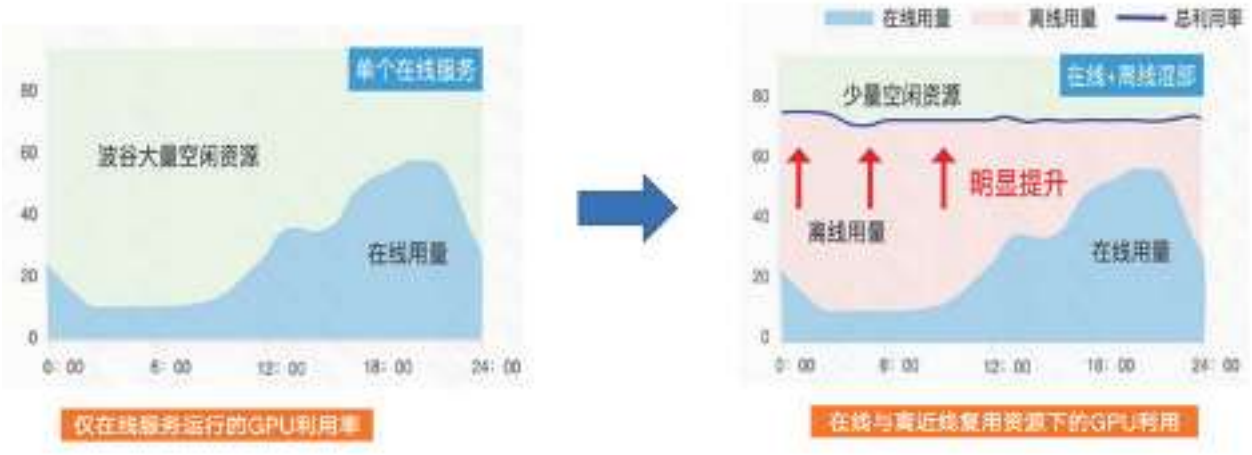


图 5-1-3　资源利用提升对比图

在资源调度层，对 GPU 进行了虚拟化，研发了 VGPU 技术，调度器会调度多个容器来共享使用一张卡提升利用率，为了避免多个容器对现存以及算力的竞争，VGPU 方案还提供了显存隔离和算力隔离。

4、实践二：计算效率优化、调度优化、故障快速恢复

为了应对单任务千卡甚至万卡的性能及稳定性的挑战，进行了很多技术的探索，从多个方面进行系统优化。

图 5-1-4　计算效率和调度优化方向

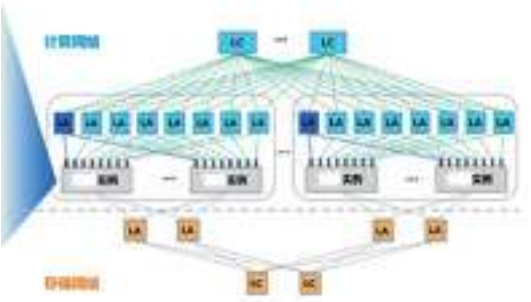


图 5-1-5　计算网络示意图

随着模型和参数的快速膨胀，模型训练过程，任务间的数据和参数通信量越来越大，网络成为了关键瓶颈，如何通过网络拓扑的感知调度增强，释放和发挥硬件网络的极限性能是非常关键的。RBAI 平台在调度上实现了交换机拓扑亲和、通信拓扑距离最小化等功能，同时通过数据亲和调度，减少不必要的跨节点数据迁移，降低网络带宽消耗。

在计算节点内，AI 平台提供了 GPU 拓扑调度来加速训练，Agent 负责 Nvlink 的拓扑发现和上报，Scheduler 根据拓扑资源以及容器资源请求给出最佳的决策。对于网络数据传输中常见的拥塞，通过定制通信和拥塞协议以及通信重叠技术，提升通信效率。AI 平台同时还在模型算法、Kernel、数据流流水线等各个维度进行了优化增强。

大模型训练周期长，目前普遍故障率比较高，尤其是训练集群规模扩大到数万个 GPU，软件和硬件故障几乎更是不可避免的。平台如果做好故障检测、快速恢复非常关键。开源的 checkpoint 和恢复方案存在效率低、耗时长。RBAI 平台针对 checkpoint 的快速恢复进行了深度优化。为了实现自动故障识别和快速恢复，在 LLM 训练中引入了一个强大的训练框架，实现了最小人为干预和对正在进行的训练任务几乎没有影响的容错能力。该训练框架提供 4 方面的核心能力，任务守护、快速故障发现、毫秒级监控、轻量级自检诊断。在实践中取得非常好的效果。

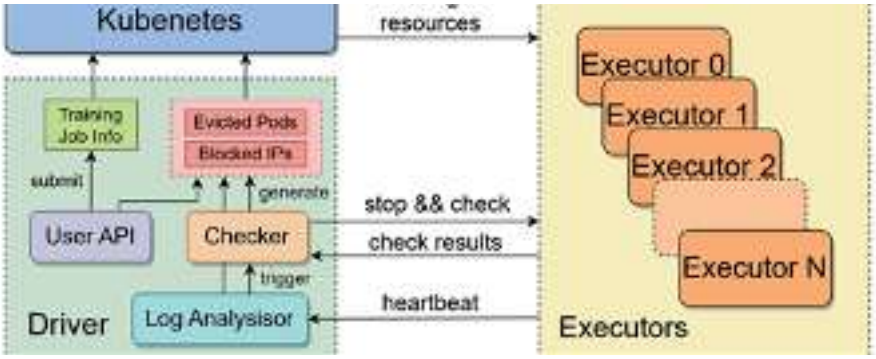


图 5-1-6　AI 业务数据流图

5、未来规划

该企业的云原生 AI 平台已深耕多年，未来会从弹性、加速、算力调度、负载调度、多集群、Workload for AI 6 个方面进行持续演进，让 AI 平台在性能、稳定性、成本上有更多的创新和突破，更好的支撑大规模分布式训练和推理场景，引领业务发展。

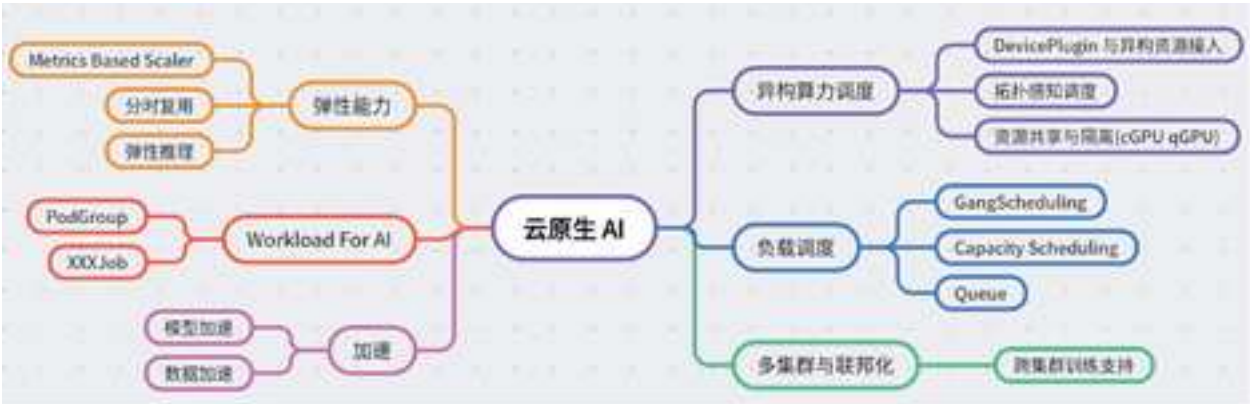


图 5-1-7　RBCNAI 演进方向

5.2 AI 解决方案提供商 FP 多场景 AI 云原生化实践

1、业务场景

内部的研发测试平台。FP 根据客户定制化的需求，赋予通用大语言模型在垂直领域的专业能力，为此，需要科学家经常对大语言模型进行微调，并评估模型效果。而过去让科学家直接虚拟机进行微调和评测的做法，经常会造成算力的浪费，从而造成任务的积压。为了解决这个问题，FP 研发了一套打榜平台，以 k8s 为底座，将所有的计算资源进行统一管理，可以随时将任务分配到空间的设备上，极大的提升了研发和测试效率。

企业决策和大模型类产品的交付。目前 FP 企业决策产品 AIOS 和大模型产品 Modelhub 都是以 kubenetes 为底层，模型产品为上层应用的模式构建起来的。如此选型主要考虑到云原生技术在集群编排领域已经成为一种事实上的行业标准，随着产品规模的增加，集群化部署已经不可避免的成为一种趋势。

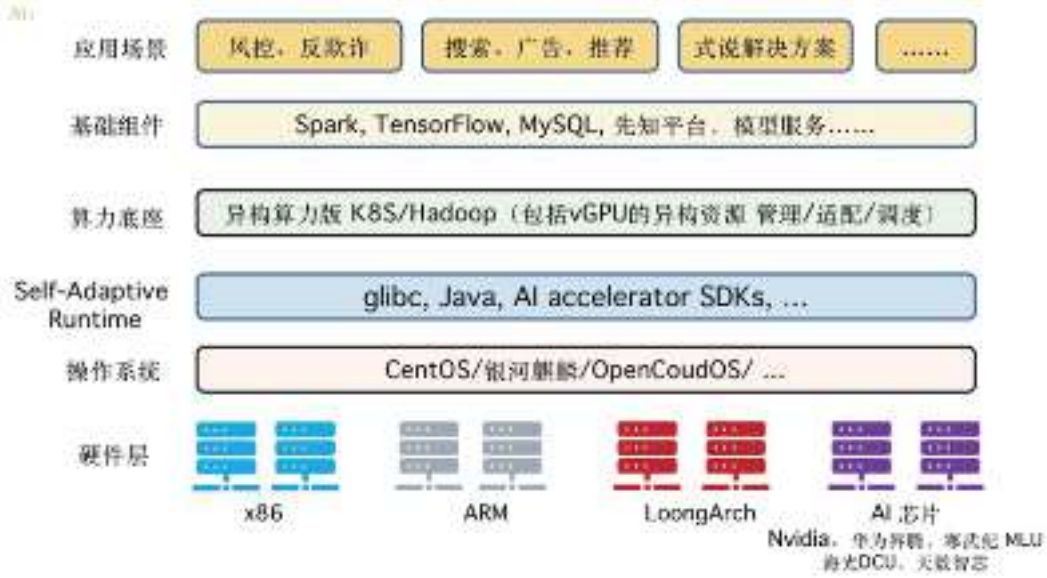


图 5-2-1　业务架构图

建设开源云原生虚拟化社区。在前两个实践的过程中，FP 也对云原生方案进行了定制化的修改，主要体现在算力底座层，FP 使用了虚拟化中间件，有效的解决了第四章节中提到的异构算力无法复用的问题。目前，FP 已经将这一部分技术开源，得到了社区的大量好评，目前 FP 正在与道客、华为云等合作伙伴一起，提交了 CNCF 基金会 sandbox 的申请。并且已经成为了 CNCF 的 landscape 项目，如下图所示：

图 5-2-2　CNCF General Orchestration

该项目主要解决了在云原生场景下如何复用第三方算力设备的问题，支持任务通过配置所需显存，算力等指标，与其他任务共享第三方算力，并保证任务的服务质量（QOS），下图可以简单描述使用该项目的效果：

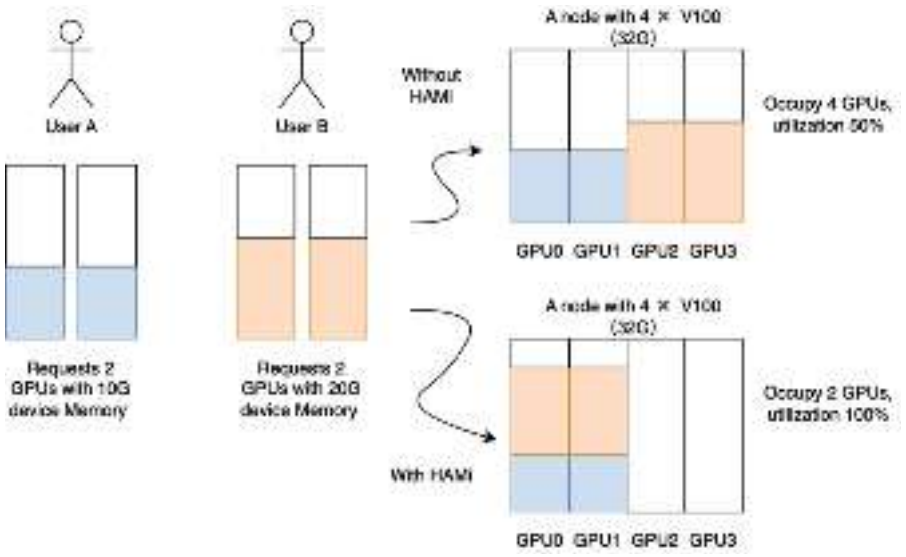
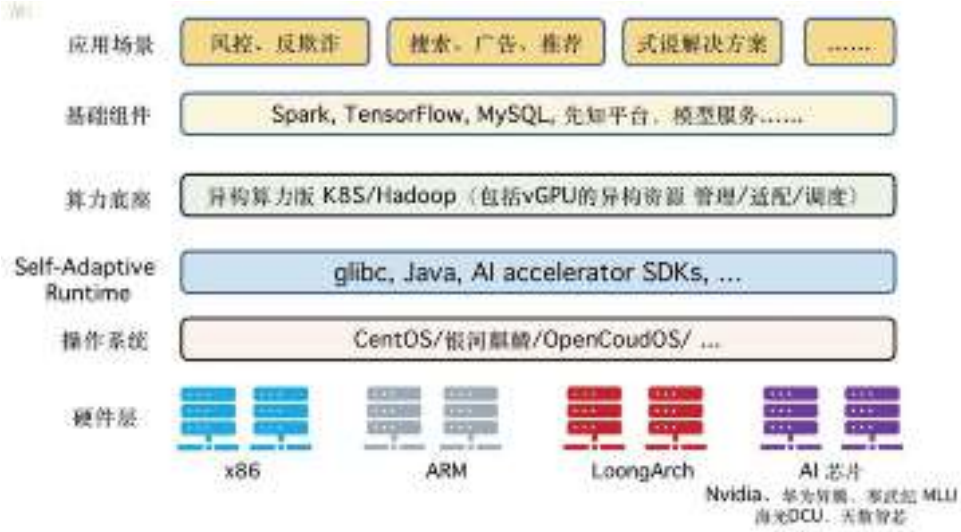


图 5--2-3　使用 HAMi 效果图

该项目目前支持的第三方算力设备包括，海光 DCU 系列，寒武纪 MLU 系列，华为昇腾 910，英伟达 GPU，天数智芯 GPU，未来也计划支持更多的算力设备。哈密瓜社区也会与其它有影响力的云原生开源项目合作（例如 volcano）, 形成更加完备的解决方案。

截止到目前，该项目下载量已经超过 10 万次，被科大讯飞，ICBC，中国联通，中国东信，H3C，新网银行，初灵股份等多家用户采纳。值得一提的是，前段实践发布的 FP 大模型推理框架 SLX-LLM 也是基于哈密瓜项目实现的性能提升。

2、技术架构



可以看出，通过云原生技术，可以将多种不同架构，不同型号的 XPU，不同操作系统，不同任务类型进行统一的资源管理，从而极大的降低部署和运维成本。

5.3 医疗科技公司 HL 云原生 AI 智能医疗实践

在当前的健康险市场中，个体风险评估变得越来越重要。传统的风险评估方法依赖于人工和经验，存在效率低、精度差的问题。随着数据量的增加和疾病种类的复杂化，传统方法难以应对现有的需求。因此，利

图 5-3-1　业务模型

用云原生人工智能（CNAI）技术进行个体风险评估成为解决这一问题的关键。CNAI 的高可用性、可扩展性和高效资源管理能力为个体风险评估提供了强大的技术支持，使其能够处理大量复杂的健康数据，并提供精准的评估结果。

1、业务场景

个体风险评估引擎，通过多源数据采集和专业疾病及风险的持续研究，定期更新数据源和升级模型，确保评估结果的准确性和时效性，实现可持续、可升级、可评估的高效精准个体风险评估。该引擎利用云原生架构和 AI 技术，能够对用户健康数据进行深度分析，提供个性化的风险评估和预测。基于 CNAI 架构，系统具有高可用性、可扩展性和高效资源管理能力，能够应对复杂多变的健康数据环境。

个体风险评估引擎应用于多个场景与多个疾病的交叉混合场景。基于个体风险评估结果，系统帮助保险公司实现精准定价，提高保费的合理性和差异化竞争力。通过自动化核保引擎，快速处理大量用户的核保请求，提升核保效率，利用 CNAI 的自动化能力优化整个业务流程。系统提供个性化的疾病管理方案，帮助用户进行健康管理，预防疾病发生，利用云原生技术实现高效数据分析和管理。

2、技术架构

个体风险评估引擎中的心血管疾病风险评估与管理的具体场景中，技术架构包括数据采集层、数据处理层、风险评估模型层、结果输出层和模型评估层。每个层次均采用了 CNAI 技术，以确保整个流程的高效性和精准性。

数据采集层

在保障数据合规的情况下，从多个数据源（电子健康档案、用户授权使用的健康数据、健康报告类数据、健康与生活习惯问卷等）获取用户授权的健康数据，通过 CNAI 技术，实现高效的数据整合和处理，确保数据的全面性和一致性。

数据处理层

对采集的数据进行清洗、预处理和标准化，以保证数据质量和一致性，并通过 CNAI 技术进行标准化处理，将不同量纲的数据转换为可比较的形式（如对血压、胆固醇进行 z-score 标准化处理，使其转换为均值为 0，标准差为 1 的分布），便于后续的特征选择和模型构建。

风险评估模型层

利用大模型能力对数据进行训练和微调，采用机器学习和深度学习算法，根据个体特征预测心血管疾病的发生概率，生成个体风险评估模型。特征选择阶段通过随机森林等机器学习算法确定哪些特征（如血压、BMI）对心血管疾病风险的预测最重要，并通过 K 均值聚类将客户基于生理指标和生活方式数据分为不同的风险群体。通过 CNAI 的分布式训练和自动调优功能，模型能够快速适应新数据，提升评估的精度和可靠性。

结果输出层

将评估结果以可视化的形式呈现，提供给保险公司和用户参考，基于云原生架构实现实时结果的输出和动态更新。CNAI 的自动化功能使得结果输出更加迅速和准确，便于用户实时获取风险评估结果。

模型评估层

使用 C 统计量和 ROC 曲线等评估指标，对模型的性能进行全面评估，通过精确率和召回率的计算，评估模型在预测心血管疾病时的准确性和覆盖范围。

图 5-3-2　模型架构图

通过利用 CNAI 技术，HL 实现了以下几方面的改进：

高效数据处理和整合：

数据采集层从多个数据源获取用户健康数据，并通过 CNAI 技术实现高效的数据整合和处理，确保了数据的全面性和一致性，极大地提高了数据处理的速度和准确性。

精准模型训练和预测：

采用华为云原生 AI 平台的分布式训练和自动调优功能，模型能够快速适应新数据，利用机器学习和深度学习算法，准确预测心血管疾病的发生概率，生成个体风险评估模型，提高了模型的精度和稳定性。

实时风险评估和动态更新：

基于 CNAI 的实时计算和推理能力使得心血管疾病风险评估能够快速响应用户数据，实现了评估结果的实时输出和动态更新，确保了评估结果的时效性和可靠性。系统不仅提供个性化的风险评估报告，还支持动态监测和定期更新，使评估结果始终保持最新状态。

实际应用中，某大型保险公司采用了远临科技的个体风险评估引擎，取得了显著成效。通过精准分析，该公司能够更好地识别高风险客户，优化风险管理流程，提高了核保效率和客户满意度。公司面临着大量客户健康数据需要分析，传统人工方法无法高效处理。通过引入远临科技的个体风险评估引擎，利用 CNAI 技术实现数据的自动化处理和分析，核保效率提高了 30%，精准定价使得保费收入增加了 15%，客户满意度提升了 20%。

3、模型开发过程的挑战

训练方面，面对海量的健康数据和复杂的模型结构，提高训练效率和模型精度是主要挑战。HL 利用云原生 AI 平台的分布式训练功能，基于平台特性采用分布式训练，将训练任务分配到多个计算节点上，显著提升了训练效率，既提高了计算资源的利用率，又减少了训练时间。利用混合精度训练技术，通过在训练过程中动态调整计算精度，保证模型精度的同时，减少计算资源消耗和训练时间。

推理方面，业务的实时性要求高且计算资源消耗量大，这对系统的响应速度和计算能力提出了严峻考

验。HL 通过模型压缩技术，如剪枝、量化和蒸馏，减小模型的尺寸和计算复杂度，提高了推理速度。同时，在推理过程中采用原生 AI 平台的推理服务，通过其提供的模型部署和管理工具，快速将训练好的模型部署到生产环境中，显著提升推理效率。

在上层框架和基础设施层面，HL 面临资源调度和管理的复杂性问题。HL 通过原生 AI 平台的统一资源资源管理自动化调度技术，实现了计算资源的自动化管理和调度，确保资源的高效利用，并在节点出现故障时自动迁移任务，保证系统的高可用性。

图 5-3-3　设施架构图