CEDFS:面向云边端跨域分布式训练的存储加速系统

刘旭昭，谭升阳，廖晓坚，霍志胜，王锦权，陈奕铭，张煜森，孙润泽，肖利民
北京航空航天大学计算机学院，北京 100191

摘要
针对云边端跨域分布式训练中广域链路时延高、带宽受限和边侧资源波动导致的存储访问开销，设计了存储加速系统 CEDFS。该系统构建统一的跨域虚拟存储空间，通过元数据批量预取与缓存、训练访问序列驱动的数据预热和状态感知的分级检查点管理，协同优化训练冷启动、样本供给及检查点保存与恢复。实验结果表明，在弱网场景下，CEDFS 的训练吞吐达到对比方案的 1.28~3.8 倍；在本地优先写入实验中，175 MB 检查点的写入阻塞时间为 0.105 s；分级恢复使训练可恢复时间缩短约 60%。该系统能够降低跨域存储访问对训练执行与故障恢复的影响。

关键词
云边端协同；跨域分布式训练；虚拟存储系统；数据预热；检查点恢复

中图分类号：TP183　　文献标志码：A

CEDFS: A Storage Acceleration System for Cloud-Edge-Device Cross-Domain Distributed Training

LIU Xuzhao, TAN Shengyang, LIAO Xiaojian, HUO Zhisheng, WANG Jinquan, CHEN Yiming, ZHANG Yusen, SUN Runze, XIAO Limin
School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract
A storage acceleration system, CEDFS, was designed to reduce the storage access overhead caused by high wide-area network latency, limited bandwidth, and fluctuating edge resources in cloud-edge-device cross-domain distributed training. A unified cross-domain virtual storage space was constructed. Batched metadata prefetching and caching, access-sequence-driven data prewarming, and state-aware tiered checkpoint management were combined to optimize training cold start, sample delivery, and checkpoint saving and recovery. Experimental results showed that CEDFS achieved training throughput of 1.28 to 3.8 times that of the baseline systems under constrained network

conditions. In the local-first writing experiment, the measured write-blocking time for a 175 MB checkpoint was 0.105 s. Tiered recovery reduced the training-resumable recovery time by approximately 60%. These results demonstrate that CEDFS reduces the impact of cross-domain storage access on training execution and failure recovery.

Key words
cloud-edge-device collaboration, cross-domain distributed training, virtual storage system, data prewarming, checkpoint recovery

0 引言

近年来，云边端协同计算快速发展，智能制造、智慧交通和电力巡检等场景中已经部署了大量分布在云侧、边侧和端侧的异构算存资源。与传统“数据回传云端、模型集中训练”相比，将分布式训练拓展到云边端跨域环境，能够利用近数据资源降低回传流量，并兼顾隐私保护、业务实时性和模型迭代效率[1-5]。

然而，在云边端跨域训练中，训练性能会从计算受限转向跨域往返受限。初步测量表明，在GPU资源和训练负载保持不变时，链路往返时延从 0.14 ms 增至 100 ms 后，基于逐次远程访问或通用缓存的方案稳态吞吐下降至 3.0~8.1 img/s；而将跨域输入/输出进行聚合、预热和异步化后，吞吐可维持在 168~189 img/s。这说明，大量细粒度远程过程调用、远程样本读取和同步检查点写入会被跨域 RTT 放大，并进入训练关键路径[6-23]。

进一步看，跨域往返放大不是某一条I/O 路径的孤立问题。训练冷启动阶段的目录遍历会决定样本索引和后续预热任务，样本供给阶段的数据预热会与检查点复制共享跨域带宽和本地缓存空间，检查点副本布局又会反向影响故障恢复时的跨域读取路径。因此，元数据、样本和检查点三类路径在云边端环境中共享拓扑状态、链路预算和缓存资源，孤立优化容易在另一条路径上制造新的瓶颈。

现有系统的设计假设与上述环境并不匹配。传统分布式文件系统和多云存储系统通常默认数据中心内低 RTT、稳定带宽和相对同构的节点环境；已有缓存和预取机制多以本地或数据中心内访问为对象，偏向被动响应已发生的访问；检查点系统则主要优化集群内持久化或故障恢复，较少把跨域 RTT、边侧副本可达性和训练前台阻塞放在同一条路径上建模[8,11,13,15,18,24-28]。当这些假设迁移到跨域训练时，统一命名空间、缓存预取和检查点管理虽然分别有效，但缺少面向训练全生命周期的存储侧协同。

基于上述观察，本文的核心判断是：云边端跨域训练的主要矛盾不是单个存储模块的吞吐不足，而是训练关键路径中的跨域 I/O 往返被反复放大；因此，需要在存储系统内部统一组织元数据、样本和检查点三条路径，把可预测的训练访问序列、链路状态和副本位置转化为协同调度依据。

为此，本文设计并实现了面向云边端跨域分布式训练的存储加速系统 CEDFS（Cloud-Edge-Device File System）。CEDFS 位于训练框架与底层物理存储之间，将云、边、端存储资源组织为统一的跨域虚拟存储空间，并通过元数据批量聚合与缓存、训练访问序列驱动的数据预热、状态感知的分级检查点管理，将跨域往返

从训练前台关键路径中前移、合并或异步化。

本文的主要贡献体现在三个方面：(1) 基于跨域弱网环境的测量结果，指出 RTT 放大、路径耦合和被动缓存失效是云边端训练存储性能退化的关键原因；(2) 提出 CEDFS 的跨域虚拟存储与存储侧协同优化设计，在不改变上层训练语义的前提下统一加速冷启动、样本供给和检查点恢复；(3) 基于真实训练任务和机制级实验验证 CEDFS 的有效性。结果表明，CEDFS 能够显著提升弱网场景下的训练吞吐，降低元数据访问与检查点写入阻塞，并稳定缩短训练可恢复时间。

1 研究背景

1.1 云边端训练存储环境

云边端跨域训练通常由三类资源共同参与。云侧节点提供大容量历史数据存储和更高可靠性的持久化能力；边侧节点更靠近业务现场，通常具备中等规模的 SSD 或 NVMe 存储与较强的本地处理能力；端侧设备持续产生图像、视频帧和传感器日志等训练数据，并提供轻量缓存或预处理能力。三类资源通过广域链路和局域链路连接，形成天然分层、异构且跨域的训练存储环境 [1-5]。

在这类环境中，训练数据的空间分布与训练状态的生命周期分布并不一致。大量原始样本在边、端侧产生，检查点的保存与恢复则受副本位置、写入方式及链路条件共同影响，目录元数据和块位置信息又必须跨域维护。若简单地将全部数据回传云端训练，会带来明显的上行带宽占用和隐私风险；若完全依赖本地边侧训练，又难以整合跨域数据和共享恢复能力。因此，云边端跨域训练天然需要统一、跨域且可感知链路状态的存储支撑。

本文采用云侧运行训练任务、边侧提供训练数据的端到端训练场景。端侧设备持续采集训练数据并就近上传至边侧，边侧节点负责数据集驻留与供给；云侧训练进程跨域读取样本，完成模型计算、梯度同步与优化器更新，并生成检查点。CEDFS 将云侧持久化存储与已有边侧存储纳入统一的跨域虚拟存储空间，由存储层按照检查点内容等级和节点状态分配保存目标，训练计算位置与检查点副本位置分别确定。

检查点副本的放置位置与复制方式由存储策略分别确定。仅在云侧保存检查点及其副本能够满足云侧训练的基本保存与恢复需求。本文采用云边统一存储部署，边侧存储作为可选副本目标，用于利用已有存储容量、分担副本存储负载，并提供可选择的恢复数据源。边侧副本无需在每次保存时一律同步完成，其创建可以采用异步复制，同时副本分布在云、边两个域，可以跨域容灾，避免单个域故障导致检查点丢失。具体副本数量和保存目标结合内容等级、节点容量与链路条件确定。

CEDFS 在 SDK 中采用异步副本复制，将首个目标的数据写入与后续云边副本传输组织为不同任务。异步执行方式与保存调用的完成条件分别确定；保存过程仍需进行必要的写入确认或等待，任务入队并不代表副本已可用于恢复。异步传输仍占用网络带宽和存储资源，并可能与训练数据读取产生竞争。本文通过分级副本配置、传输调度与恢复源选择，降低云边存储中的检查点写入与恢复开销。

1.2 面向训练加速的存储路径挑战

从训练过程看，存储系统主要沿三条路径影响训练效率。第一条是冷启动元数据路径。以 ImageFolder 一类数据集为例，训练框架在启动阶段需要完成目录递归遍历和文件属性查询，小文件数量越多，跨域元数据 RPC 次数越高，冷启动耗时越容易被放大[26-32]。第二条是训练样本读取路径。DataLoader 在每个 epoch 中持续按采样顺序读取文件，如果样本位于远端节点，带宽、RTT 和丢包都会直接进入 batch 的关键路径[6-12]。第三条是检查点写入与恢复路径。检查点包含模型权重、优化器状态和分片映射等内容，若写入路径设计不当，会导致训练主循环长期阻塞；若恢复副本布局不合理，则故障后恢复时间难以控制[13-23]。

更重要的是，这三条路径并非彼此孤立。元数据路径影响样本访问序列构建与预热任务生成，数据预热又与检查点恢复共享带宽和缓存资源，检查点副本放置还会反向影响跨域存储负载。因此，云边端跨域训练更需要从系统层面统一组织这些路径，而不是依赖若干互不协调的局部优化。

1.3 相关工作

传统分布式文件系统和多云存储系统围绕统一命名空间、元数据扩展和缓存组织开展了大量研究。早期系统如 NFS、AFS、Sprite 和 Gfarm 奠定了跨节点共享访问基础[24,33-35]；CephFS、HDFS 和 Tachyon 等存储系统进一步支撑大规模数据管理[25,32,36-39]；IndexFS、LocoFS、HopsFS、InfiniFS、CFS 和 SingularFS 等工作则重点优化元数据扩展能力[26-31]。这些工作为跨节点数据管理与共享访问奠定了基础，但总体上更面向集群内相对稳定的互联环境。

面向云边端的跨域存储研究进一步关注路径组织、边侧索引和协作存储，例如协作边缘存储、PathFS 以及边缘环境下的分布式文件系统等[1-5]，PhatKV、EDIndex 和 COIN 等工作则关注边侧索引、KV 元数据引擎和非结构化数据查询[40-42]。针对训练数据访问，已有工作主要从缓存和预取两方面进行优化，例如 SOLAR、SHADE、Lobster、DIESEL、Fluid、CrossPrefetch 和 Clairvoyant Prefetching 等[6-12]；其中，Quiver 面向 GPU 集群内的深度学习训练存储缓存，采用命中率模型与短周期 I/O 观测驱动缓存决策[43]；该类方案面向同域高速互联环境，不涉及跨域 WAN 下元数据、样本与检查点三条 I/O 路径的协同调度，且未公开可复现实现，难以在本文跨域弱网环境中直接部署对照；本文 3.1 节中的通用缓存基线（如 JuiceFS-cached）在缓存受限场景（图 9）下已覆盖了类似的基于命中率优化与本地缓存的行为。针对检查点存储，研究则主要围绕异步持久化、恢复优化和检查点压缩展开[13-23]。总体来看，现有工作尚缺少一个同时覆盖元数据访问、数据供给和检查点恢复的云边端跨域训练存储系统。

明龙等围绕 Hadoop 生态系统研究了跨层级存储与计算协同及可信调度策略[44]；SpanTrain 面向云边端异构设备实现了跨域分布式模型训练[45]。这些研究分别从系统协同优化和跨域训练执行角度提供了参考。CEDFS 进一步研究训练过程中元数据访问、样本供给和检查点恢复的存储路径优化。

2 CEDFS 系统设计

2.1 CEDFS 概览

针对上述问题，本文设计并实现了 CEDFS。CEDFS 位于训练框架与底层物理存储之间，向上通过 FUSE 和 Python SDK 提供统一文件访问与训练 I/O 接口，向下管理云、边、端节点中的元数据、数据块、副本和缓存资源。与训练框架不同，CEDFS 不负责模型计算、梯度同步和优化器更新，而是面向训练过程中的存储访问路径进行加速。

整体来看，CEDFS 主要包含客户端、元数据服务器、数据服务器以及状态感知与调度组件。客户端负责统一挂载、缓存命中、预取调度和检查点读写接入；元数据服务器负责命名空间维护、路径解析、跨域路由、批量元数据查询和节点状态汇聚；数据服务器负责数据块存储、跨域块读写、副本创建和运行指标上报。系统运行时主要沿冷启动元数据路径、训练样本读取路径和检查点写入恢复路径工作，系统总体架构如图 1 所示。

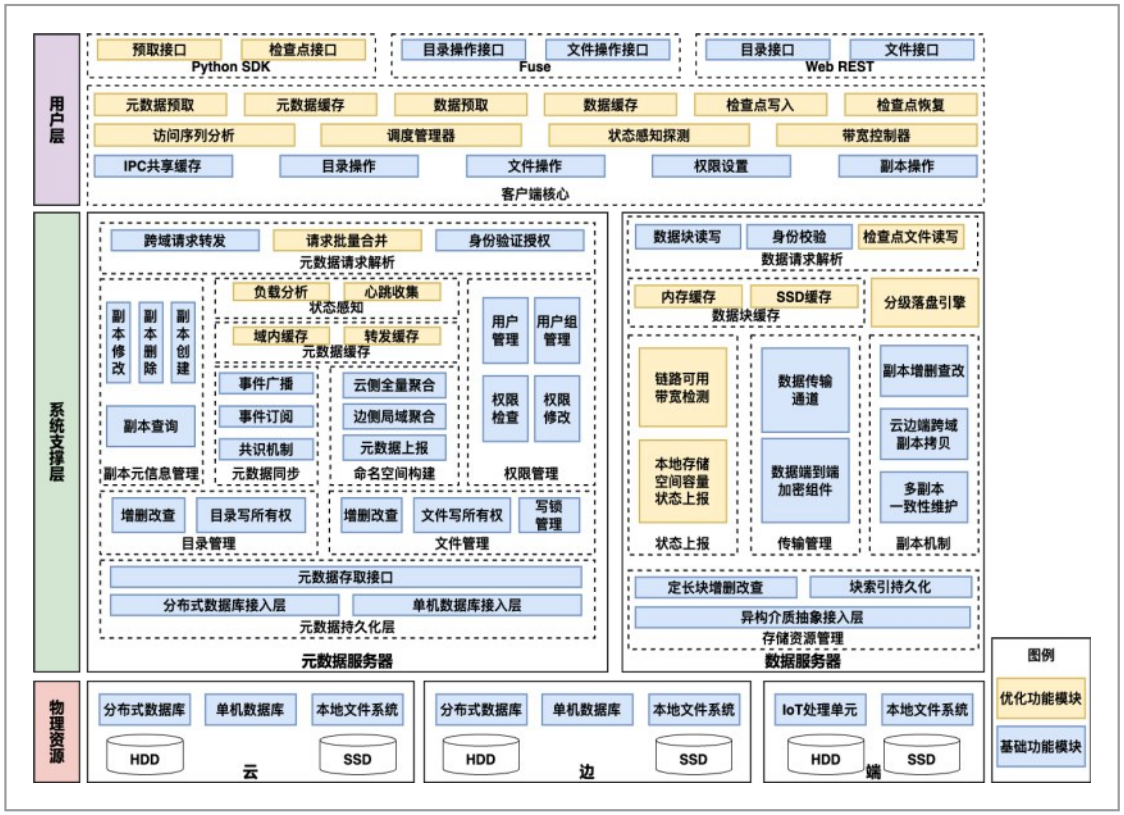


图 1 CEDFS 系统总体架构

从执行路径看，CEDFS 在训练前主要承担目录遍历和元数据归并，在训练中主要承担远端样本搬运和本地缓存命中，在训练容错阶段则负责检查点分片的分级落

盘、异步复制与恢复源选择。这样，训练框架仍保持原有的数据集、DataLoader和检查点调用语义，而跨域环境中的主要I/O复杂性被吸收到存储系统内部。

2.2 跨域虚拟存储空间

CEDFS通过层级目录树组织跨域虚拟存储空间。云侧元数据服务器维护全局目录入口，边侧元数据服务器向上注册各自管理的子树，每个元数据服务器只直接管理本域路径；当客户端访问非本域路径时，请求依据目录树和路由表被转发到目标域处理。元数据采用inode与目录项映射组织，底层由KV存储引擎完成属性与路径索引持久化。控制面基于gRPC承载路径解析、块分配、节点注册和状态上报，数据面基于QUIC完成高吞吐块传输。这样，统一命名空间既提供位置透明访问，也为后续三类加速机制提供拓扑与状态基础[24-28,38]，跨域存储空间部署方式如图2所示。

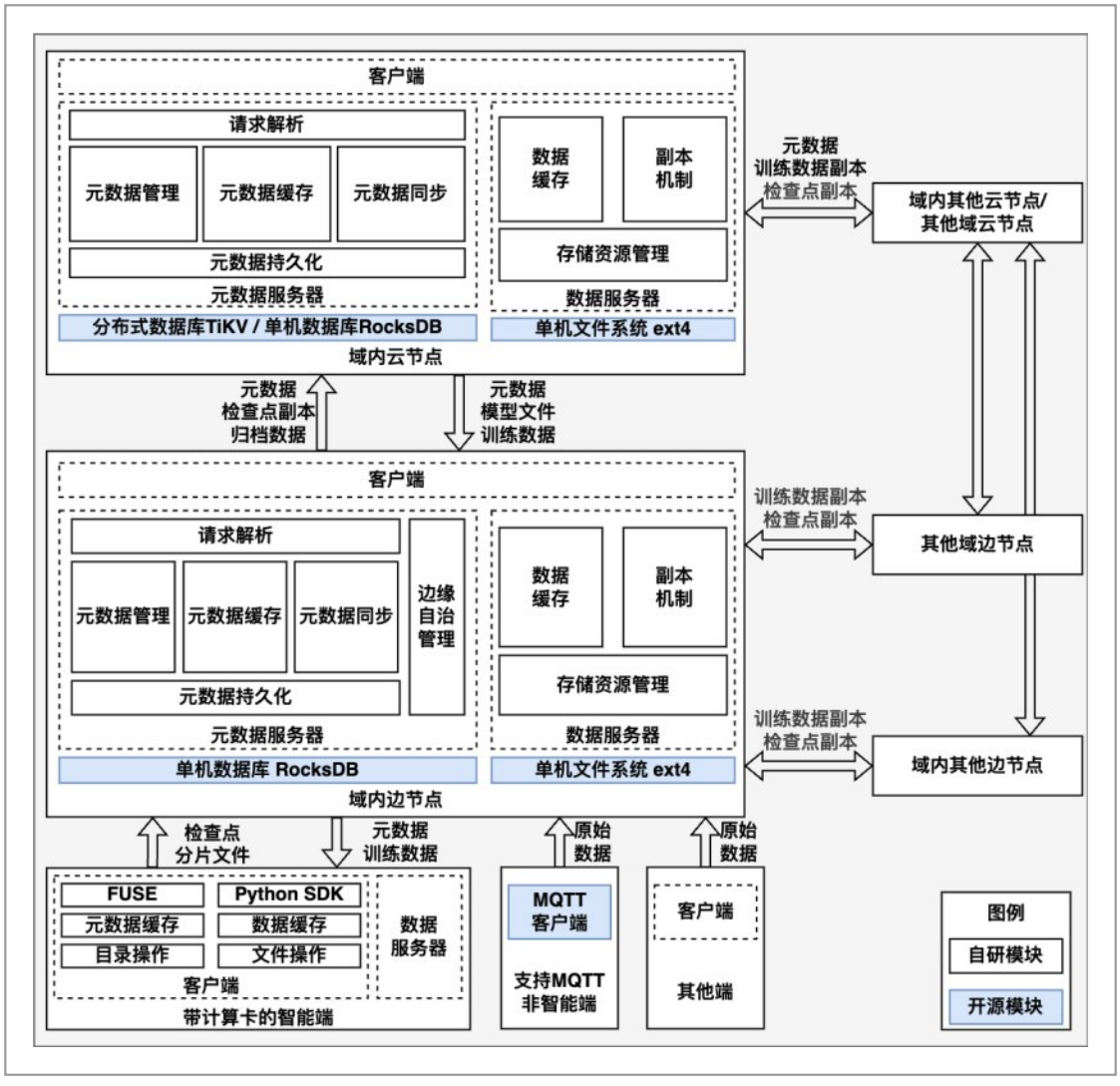


图2 云边端跨域存储空间部署

2.3 元数据批量预取与数据预热

在冷启动阶段，CEDFS 通过元数据批量预取缩短目录遍历时间。训练任务初始化数据集时，客户端递归预取引擎根据目录结构发起单目录或批量目录读取请求，将大量细粒度跨域 RPC 压缩为少量聚合请求。客户端维护内存、磁盘和 IPC 共享缓存，服务端维护域内元数据缓存和跨域转发缓存，使首次遍历得到的目录结构、文件属性和路径归属信息能够在后续 epoch 中复用[26-31]。

对于同一遍历层级内的目录路径集合 $P_{\text{level}}=\{p_1,p_2,\ldots,p_l\}$，批量读取模块按照批次大小 β 将其划分为多个批次块：$B_1,B_2,\ldots,B_{\lceil l/\beta\rceil}$

$$B_j=\left\{p_{(j-1)\cdot\beta+1},p_{(j-1)\cdot\beta+2},\ldots,p_{\min(j\cdot\beta,l)}\right\},\quad j=1,2,\ldots,\lceil l/\beta\rceil \tag{1}$$

其中，B_j 表示第 j 个批量请求块，l 为当前层级目录数量，β 为每次 BatchReadDir RPC 最多包含的目录数。该分块策略将原本 l 次独立 RPC 调用压缩为 $\lceil l/\beta\rceil$ 次批量请求，从而减少跨域往返次数。元数据批量预取与缓存方法的整体架构如图 3 所示。

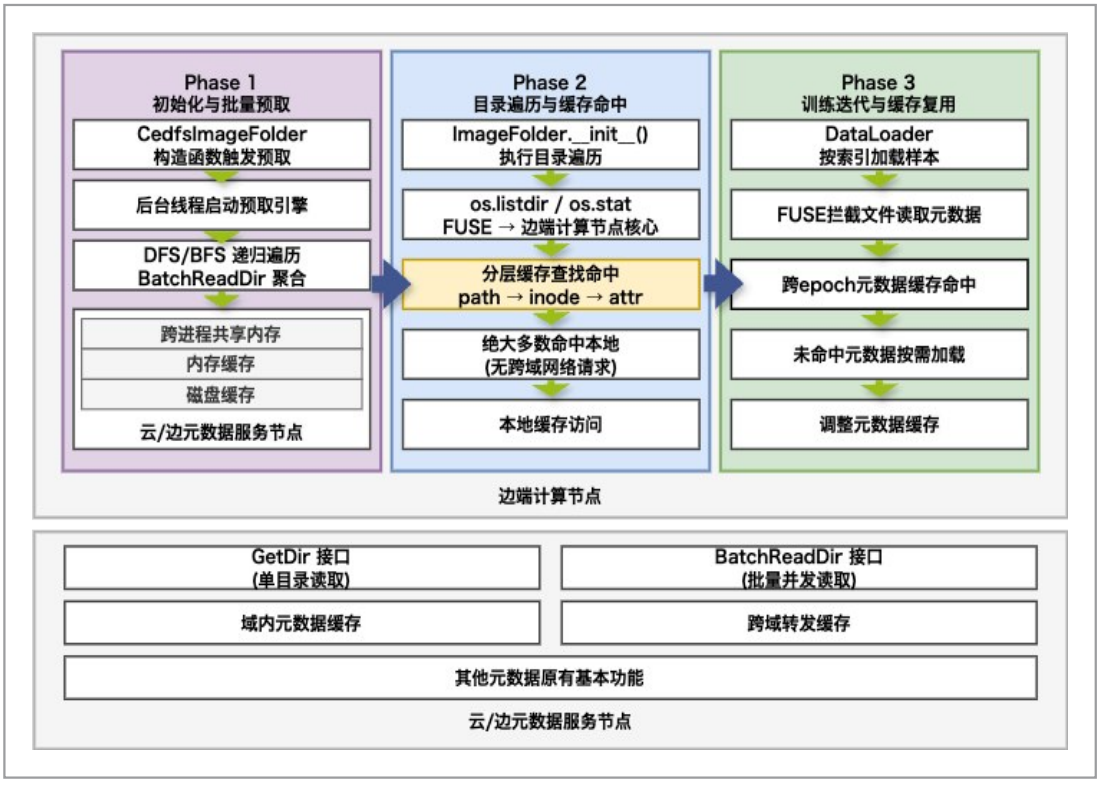


图 3　元数据批量预取与缓存方法整体架构

批量预取对训练路径的影响如下：元数据批量预取将目录遍历中的多次跨域往返合并为少量聚合请求，并把首次遍历得到的目录结构、文件属性与路径归属信息

载入多级缓存，在训练启动及后续 epoch 中持续复用，为训练读路径的样本定位与数据预热提供元数据基础；针对首批样本的就绪时延，CEDFS 将聚合元数据请求与样本数据块的拉取进行流水化执行，训练框架发起数据读取时，目录遍历请求与样本访问序列一并提交，首批数据块的传输与元数据响应在时间上重叠，因此批量块大小的变化不会显著推迟首批样本的就绪时间，其引入的额外阻塞被限制在约一个往返时延以内，且增大聚合粒度可显著缩短冷启动遍历与“启动→首批样本就绪”时延，量化结果见 3.2.2 节表 6 与图 13。

除批量聚合外，CEDFS 还通过客户端与服务端协同缓存延长预取结果的有效期。客户端内存缓存服务于低延迟命中，磁盘缓存服务于跨轮次复用，IPC 共享缓存服务于同节点多进程共享；服务端则利用域内元数据缓存和转发缓存减少本地 KV 查询和跨域转发次数。这样，训练首轮建立的目录视图能够直接支撑后续 epoch 的样本定位和访问序列生成。

在训练执行阶段，CEDFS 利用可预测的样本访问序列进行数据预热。系统通过 Python 接口获取各 rank 的文件访问序列，预取引擎结合访问顺序、数据所在域和文件大小生成任务，采用滑动窗口合并相邻读取，并按带宽配额和缓存水位执行异步调度。预取线程通过 QUIC 拉取远端数据块并写入本地两级缓存，从而将更多跨域读取转化为本地命中[16-12]。

对于当前 epoch 的访问序列 $\sigma_e=\{f_0,f_1,\ldots,f_{N_s-1}\}$，CEDFS 为第 i 个文件构造预取优先级：

$$priority(f_i)=(N_s-i)\times w_d(f_i)\times\left(1+\beta\cdot\log_2\left(1+\frac{size(f_i)}{B_{base}}\right)\right) \quad (2)$$

其中，N_s-i 为基础顺序优先级，$w_d(f_i)$ 为文件 f_i 所在数据节点的域延迟，$size(f_i)$ 为文件大小，B_{base} 为归一化基准大小，β 为文件大小权重系数。该式的目标不是改变训练访问顺序，而是在保持访问次序基本一致的前提下，让远端高时延、大体积文件更早开始搬运。

为了避免训练启动时缓存准备不足，调度器维护 warmup 覆盖率：

$$\eta=\frac{\text{completed}_{\text{tasks}}}{N_s} \quad (3)$$

其中，$\text{completed}_{\text{tasks}}$ 表示当前预热轮次已完成的预取任务数，N_s 表示样本总数。当覆盖率 η 达到目标阈值或等待时间超过上限时，训练启动；若未达标，则系统以当前缓存状态放行训练，并让未命中数据退化为同步跨域访问。数据预热技术的整体架构如图 4 所示。

在工程实现上，CEDFS 将“序列注入、任务合并、带宽控制、缓存写入”串成一条异步管线。当前台 Worker 访问样本时，若目标数据已命中本地缓存，则直接返回；若尚未命中，则退化为常规跨域拉取。随着 epoch 推进，更多远端访问会被逐步转化为本地命中，从而降低训练过程中 batch 等待数据的时间。

2.4 检查点分级写入与恢复

在训练容错阶段，CEDFS 通过 PyTorch DCP 的 StorageWriter 和 StorageReader 接入检查点保存与加载流程，将检查点分片的写入、复制和恢复统

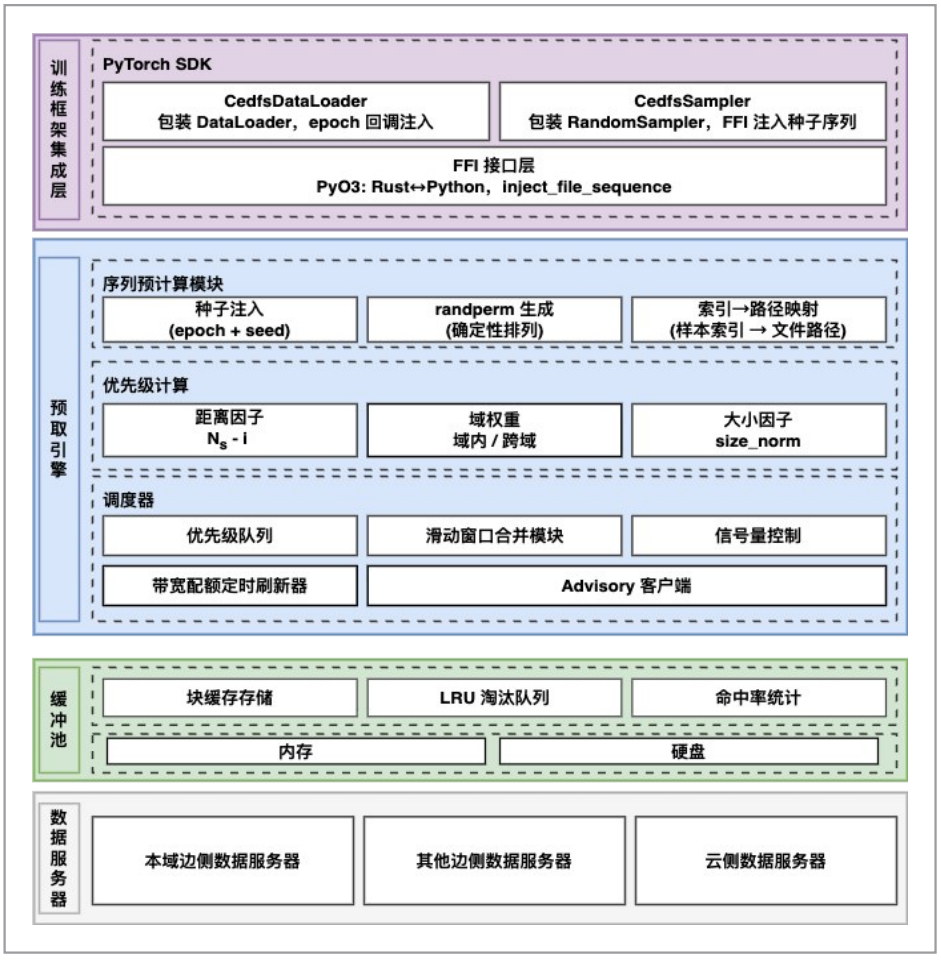


图 4　基于异步预取调度的数据预热技术整体架构

一下沉到存储系统内部。系统持续跟踪各节点的容量、读写吞吐、网络可达性和拓扑距离，为副本放置和恢复源选择提供依据[13-23]。检查点分级写入技术的整体架构如图 5 所示。

对于当前写入节点 n_s 和候选目标节点 n_t，CEDFS 先用分层距离描述副本放置的拓扑代价：

$$\mathrm{dist}(n_s,n_t)=\begin{cases}0 & n_t=n_s\\ 1 & n_t\neq n_s\wedge\tau(n_t)=\mathrm{Edge}\wedge z(n_t)=z(n_s)\\ 2 & \tau(n_t)=\mathrm{Edge}\wedge z(n_t)\neq z(n_s)\\ 3 & \tau(n_t)=\mathrm{Cloud}\end{cases}\tag{4}$$

其中，$\tau(n)$ 表示节点类型，取值为 Edge 或 Cloud，$z(n)$ 表示节点所属域标识。该定义将本地、域内、跨域和云端四类目标统一映射到同一距离空间，便于后续副本放置和恢复排序使用。

CEDFS 将 .metadata 全局分片映射、模型权重、通用训练参数和优化器状态依次标记为 Critical、High、Medium 和 Low，用于副本配置与恢复排序。图 5 概括分级执行策略：同步写入对应选定目标的写入确认，异步写入与复制对应后台调度，不要求完整检查点及其全部边侧副本一律同步完成。SDK 先调用首个目标的分片写入接口，再通过异步任务创建其他边

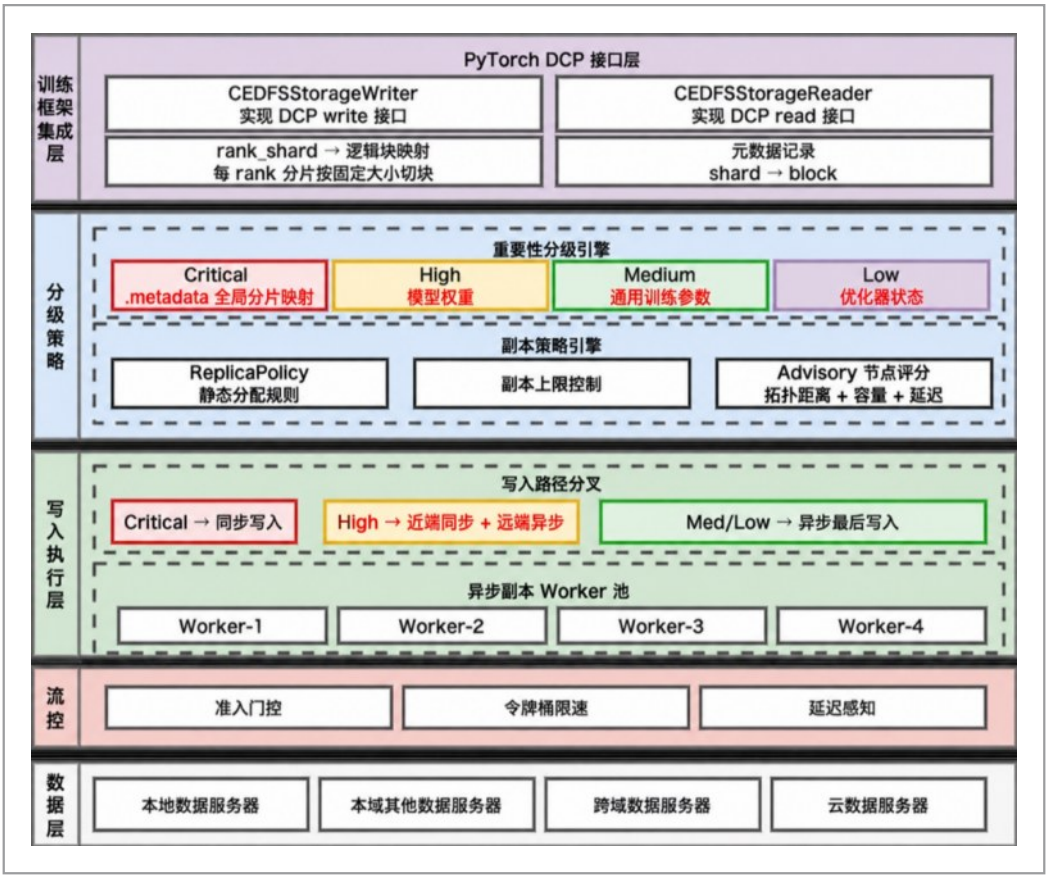


图 5 基于状态感知分配的检查点分级写入技术整体架构

侧或云侧副本，元信息在保存收尾阶段处理。该路径仍包含写入确认与复制任务等待。边侧节点可作为首个目标或后续副本目标，后台任务通过并发传输与带宽控制组织；恢复时结合可达性、拓扑距离和探测结果选择具有所需数据的副本。

在恢复源选择中，CEDFS 使用延迟追踪器记录各存储源的历史访问延迟。设每个源维护最近 w 条延迟记录，则实时延迟估计采用近期加权平均：

$$L(n_s)=\frac{\sum_{i=1}^{w} w_{\text{recency}}^{w-i}\cdot l_i}{\sum_{i=1}^{w} w_{\text{recency}}^{w-i}} \tag{5}$$

其中，l_i 为第 i 条延迟记录(按时间升序)，$w_{recency}$ 为近期权重系数。这里的 n_s 表示被记录访问延迟的存储源节点。该延迟估计与分层距离共同用于恢复阶段的源选择；对于大体积检查点分片，系统会先发送小规模探测块，再更新延迟记录并确定后续全量读取的数据源。检查点分级写入与恢复流程如图 6 所示。

图 6 示意上层将保存任务交由后台调度时，训练与存储写入可重叠执行的时序。训练继续时点由保存接口的写入确认与等待条件决定，SDK 内部异步复制不代表整个保存调用均与训练重叠；图中并行区间不对应图 14 的计时范围。尚未写完所需状态的副本不能用于恢复当前版本；若其他可用源也缺少所需状态，则选择已完成的较早检查点。

3 性能实验

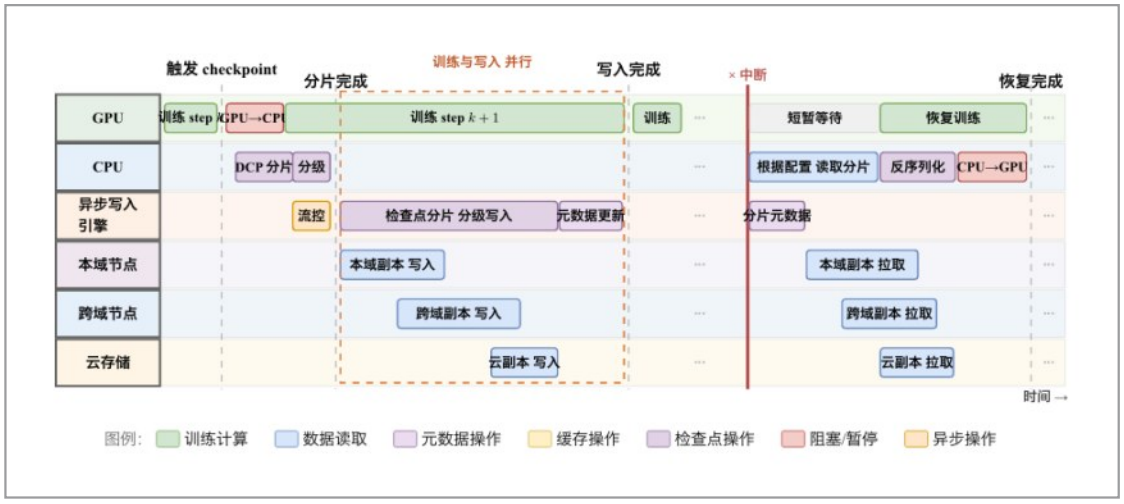


图 6　检查点分级写入与恢复流程

本节从端到端训练性能对比、机制效果与稳健性对比 2 个方面，阐述 CEDFS 在云边端跨域训练存储加速中的性能表现。

3.1 端到端训练性能评估

本节在三节点物理弱网环境中评估 CEDFS 对训练主流程的端到端加速效果，重点考察弱带宽、缓存受限和 RTT 敏感性等训练执行阶段能够直接感知的开销。该环境用于回答 CEDFS 是否能够提升真实训练吞吐。

(1)　实验环境与参数配置

端到端训练实验在 3 台物理服务器上完成，硬件与软件环境如表 1 所示。

表 1　端到端实验的硬件与软件环境

配置项	参数
节点数量	3 台物理服务器
处理器	2× Intel Xeon Gold 6330
内存	512GiB DDR4
GPU	n33: 8 × 天垓150(64 GiB)；n192/n197: 8× 天垓100(32 GiB)
存储	NVMe SSD 3.6TiB
操作系统	Ubuntu 20.04.6 LTS
Python	3.10
PyTorch	2.7.1

三节点构成云边端三域拓扑，云节点为 n33，远端训练数据默认放置在边缘节点 n192 和 n197，保证训练数据访问稳定经过跨域链路。需要说明的是，该实验部署中训练任务运行于云端节点、训练数据存储于边缘节点，数据流方向为云端从边

缘拉取；由于云边链路两端施加同等网络约束，各对比方案经历相同的软件栈和链路条件，因此该拓扑不影响相对性能比较。节点部署和默认网络模拟参数如表 2 所示。

表 2 端到端实验的默认网络模拟参数

链路	单边延迟/ms	RTT/ms	带宽/(Mbit·s^{-1})	模拟场景
n33 ↔ n192	2 ms ± 1 ms	~5 ms	80 Mbit/s	Cloud ↔ Edge-A
n33 ↔ n197	2 ms ± 1 ms	~5 ms	80 Mbit/s	Cloud ↔ Edge-B
n192 ↔ n197	5 ms ± 2 ms	~10 ms	40 Mbit/s	跨边缘域

训练节点为 n33，远端训练数据默认放置在 n192 和 n197，并通过 tc/netem 在云边链路上注入带宽、RTT、抖动和丢包约束，默认 RTT 约 5 ms、带宽 80 Mbit/s。

表 3 给出了三节点实验的对比方案、训练负载和关键参数。端到端训练实验主要比较 CEDFS-full、CEDFS-direct、NFS [2 4]、JuiceFS 和 JuiceFS-cached，其中 CEDFS-full 启用缓存、预热和检查点优化，CEDFS-direct 关闭这些优化以体现系统机制带来的差异。

表 3 实验的对比方案与关键参数

项目	配置
存储系统	CEDFS-full、CEDFS-direct、NFS、JuiceFS、JuiceFS-cached
模型	ResNet-50
数据集	Imagenette2-320(367 MB)/ Imagenette2(1.5 GB)
分布式训练(DDP)	world_size=2
batch_size	64
训练轮数	5
优化器	SGD
随机种子	42
CEDFS 缓存	256MiB 内存 + 256MiB 磁盘
JuiceFS 缓存	256MiB 内存 + 256MiB 磁盘

实验结果分析

本节重点保留弱带宽、缓存受限和 RTT 敏感性三类代表性场景，分别对应“跨域链路受限”“本地缓存不足”和“跨域往返时延增大”三种常见瓶颈。

首先分析弱带宽条件下的训练吞吐。为隔离带宽因素，实验固定 RTT 约 4 ms，扫描 10~500 Mbit/s 的链路带宽，并比较各方案的稳态吞吐。该组实验使用 Imagenette2-320，数据集规模约 367 MB，小于 CEDFS-full 与 JuiceFS-cached 的本地缓存预算，因此主要反映访

问路径和预热机制对训练主流程的影响。

图 7 所示为不同存储方案在不同带宽条件下的稳态吞吐量。可以看出，在 10~500 Mbit/s 带宽范围内，CEDFS-full 的稳态吞吐为 118.8~214.1 img/s，始终优于其余对照方案。具体而言，在 10 Mbit/s 时，CEDFS-full 的吞吐量分别达到 CEDFS-direct、NFS 和 JuiceFS-cached 的 3.2 倍、3.8 倍和 1.28 倍；在 40 Mbit/s 时，仍分别保持 2.7 倍、3.8 倍和 1.69 倍的领先。进一步看曲线形态，当带宽提升到 40 Mbit/s 以上后，CEDFS-full 基本进入平台区，说明此时训练关键路径已不再主要受链路带宽限制。

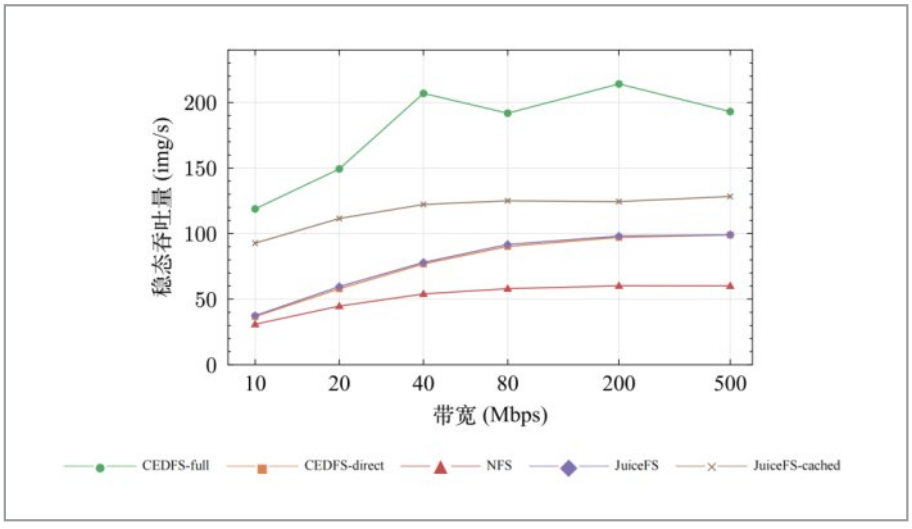


图 7　各存储方案稳态吞吐量随带宽变化

冷启动结果与上述趋势一致，如图 8 所示。CEDFS-full 首个 epoch 的总数据加载时间随带宽从 10 Mbit/s 提升到 500 Mbit/s 时，由 120.0 s 降至 35.7 s；CEDFS-direct、NFS 和 JuiceFS-cached 分别为 128.7~45.6 s、151.9~76.4 s 和 173.6~87.4 s。该结果表明，即使承担首轮预取和缓存建立开销，CEDFS 仍保持了较优的冷启动性能。

进一步考察缓存受限场景。实验将数据集替换为 Imagenette2 全分辨率版本，在其余配置不变的条件下重新扫描 10~500 Mbit/s 带宽，以观察当数据规模逼近或超过缓存预算时，各方案的退化行为。

如图 9 所示，在缓存受限条件下，CEDFS-full 在 10 Mbit/s、20 Mbit/s 和 40 Mbit/s 时的稳态吞吐分别为 32.5 img/s、69.8 img/s 和 78.6 img/s，仍明显高于 NFS 与 JuiceFS-cached；即使在 500 Mbit/s 下，CEDFS-full 仍达到 130.5 img/s，高于 NFS 的 51.9 img/s 和 JuiceFS-cached 的 81.5 img/s。与图 7 对应场景相比，本组所有方案的绝对吞吐都明显下降，说明数据集规模一旦逼近或超过缓存预算，链路搬运开销会显著增大；但 CEDFS-full 仍能通过更有针对性的预热与缓存分配保持领先。

在 RTT 敏感性实验中，RTT 从 0.14 ms 增加到 100 ms 时，CEDFS-full 的稳态吞吐仍保持在 168~189 img/s；其

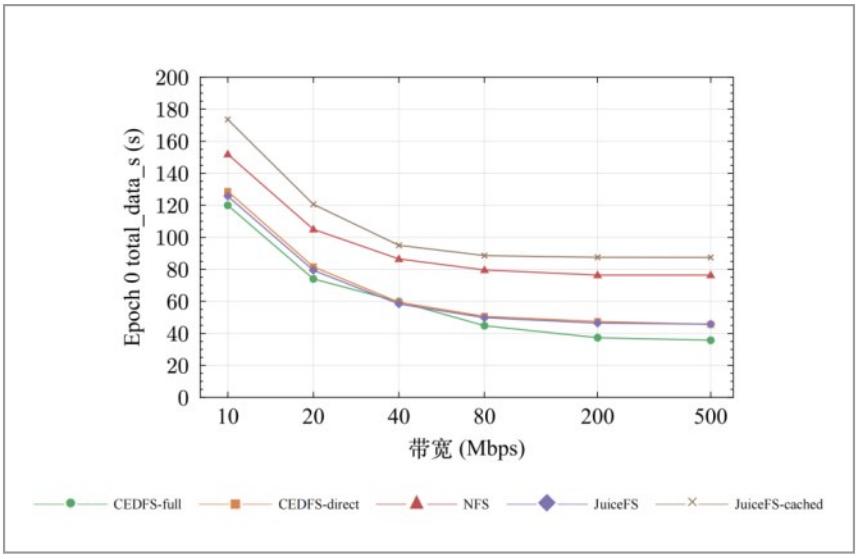


图 8 首个 epoch 总数据加载时间随带宽变化

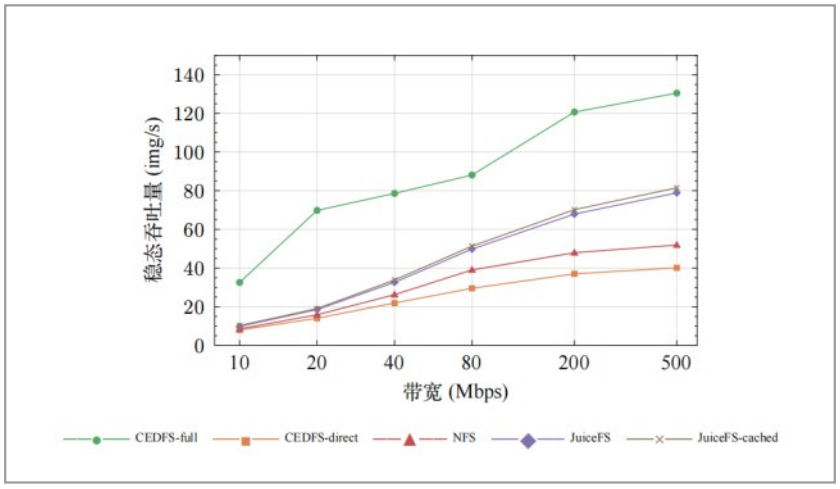


图 9 缓存受限下稳态吞吐量随带宽变化

余方案在 RTT=100 ms 时下降到 3.0~8.1 img/s，结果如图 10 所示。该结果说明，CEDFS 对跨域往返时延的敏感性明显低于逐次远程读取方案。

为验证系统在主流大语言模型训练负载（含多模态）下的泛化性，本文补充了 Qwen2-VL-7B 多模态模型端到端实验：训练数据采用 COCO 图文数据集（图像统一缩放到 448×448 的 JPEG，配以 token 序列文件），全程 bf16 混合精度并启用梯度检查点。多模态训练的数据管线同时包含大文件（图像）与小文件（文本样本），且图像与文本均需从边侧跨域拉取；网络条件采用与 3.2 节集群实验一致的云边链路（80 Mbit/s、RTT≈50 ms，对比方案同表 3），以在弱网跨域环境下放大数据路径在训练循环中的占比、使存储方案间的差异可被端到端观测。该负载下

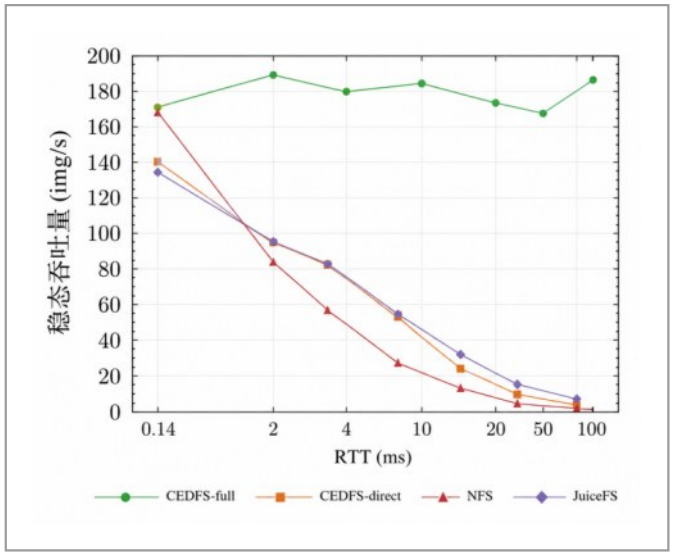


图 10　稳态吞吐量随 RTT 变化

训练计算每步数秒，而跨域数据等待构成可观开销（其他方案实测数据等待占比约 40%~50%），因此本组实验重点考察 CEDFS 在多模态训练读路径上能否通过元数据批量预取与数据预热机制同时压缩图像与文本的跨域读取等待、提升端到端吞吐；结果如表 4 所示。

表 4　Qwen2-VL-7B 多模态训练负载下的训练吞吐与数据路径开销对比

方案	训练吞吐/(tok·s^{-1})	TTFB/ms	平均数据等待/ms	数据等待占比/%
CEDFS-full	795	1346	77.2	1.70
CEDFS-direct	444	3924	3719.2	45.61
NFS	495	3186	2896.0	39.62
JuiceFS	400	4867	4664.4	51.51
JuiceFS-cached	396	5202	4697.8	51.38

如表 4 所示，在 Qwen2-VL-7B 多模态负载下，跨域图文读取构成显著的数据路径开销：无预热的直连 CEDFS、NFS 与 JuiceFS 的每步平均数据等待达 2.9~4.7 s（占训练循环的 39.6%~51.5%），而 CEDFS-full 通过元数据批量预取与数据预热将每步数据等待压缩至仅 77.2 ms（占比 1.70%）；相应地，CEDFS-full 的每进程训练吞吐达 795 tok/s（4 进程并行时系统总吞吐约为表中数值的 4 倍），较直连 CEDFS（444 tok/s）与 NFS（495 tok/s）分别提升约 79% 与 61%，较 JuiceFS（400 tok/s）提升约 99%；首批样本就绪时延（TTFB）亦由对比方案的 3186~5202 ms 降至 CEDFS-full 的 1346 ms。此外，JuiceFS-cached 与无缓存

JuiceFS 的吞吐接近（396 vs 400 tok/s），原因是单 epoch 训练中每个样本文件仅被读取一次、缓存命中率有限。上述结果说明 CEDFS 的元数据批量预取、数据预热与缓存机制在 7B 参数量级的多模态负载下，能够同时压缩图像（大文件）与文本（小文件）的跨域读取等待、带来显著的端到端吞吐提升，且不会引入额外关键路径负担。

3.2 关键存储机制性能分析

本节在 ECS 集群上进行机制级实验，重点分析元数据批量预取、检查点分级写入、分级恢复和故障恢复路径选择的效果。该环境用于解释端到端收益的来源，其目的不是替代三节点物理弱网环境的训练性能结论，而是在更多元数据服务、数据服务和客户端角色并存的可控拓扑中观察机制本身。

3.2.1 实验环境与配置

集群的硬件、软件、部署拓扑和网络模拟参数如表 5 所示。实验采用三域部署模型模拟云边端跨域环境：云端部署全局元数据根服务器和数据服务器，边缘域部署本地元数据子服务器、数据服务器和训练客户端。跨域网络延迟通过 tc/netem 注入，带宽通过 HTB 队列规则进行整形。

表 5 实验的部署拓扑与网络配置

配置项	参数
节点规模	10 个 ECS 节点
硬件配置	云端 8vCPU/16GB，边缘元数据节点 4vCPU/8GB，训练客户端节点 2vCPU/2~4GB；
存储配置	40 GB ESSD
软件环境	Ubuntu 24.04.4 LTS，Linux 6.8.0，Rust 1.82.0 (nightly)，Python 3.12.3，PyTorch 2.10.0+cu128
Cloud 域	1 个节点，部署 Cloud Meta 和 Cloud Data
Edge-A 域	4 个节点，部署 Edge Meta-A、Data-A 和多个 Client
Edge-B 域	5 个节点，部署 Edge Meta-B、Data-B 和多个 Client
网络配置	域内 LAN：1 ms、400 Mbit/s；Cloud-Edge：25 ms ± 10 ms、80 Mbit/s；Edge-A 至 Edge-B：50 ms ± 20 ms、40 Mbit/s；

3.2.2 实验结果分析

首先考察元数据批量预取与缓存的效果。元数据实验采用 Full 和 Bypass 两种配置进行对比，其中 Full 启用批量目录预取、BFS 遍历和多级缓存，批量块大小设为 50，内存元数据缓存配额为 512 MB；Bypass 关闭预取和缓存，代表逐条跨域请求模式[26-31]。实验使用 syn-100c-50f 和 syn-1000c-50f 两组合成数据集，分别对应 5 000 和 50 000 个文件规模。

训练任务在启动阶段需要遍历目录、

读取文件属性并构建样本索引，如果这些操作逐条跨域发送，网络往返会直接放大冷启动开销。在 syn-1000c-50f 数据集上，Full 模式在首轮遍历时耗时 5 110 ms，从第 1 个 epoch 起稳定在约 430 ms，而 Bypass 模式始终维持在约 89 s，如图 11 所示。该结果表明，首轮批量预取完成后，目录条目和文件属性已经驻留在本地缓存中，后续轮次可以直接命中缓存而无需重复跨域请求。

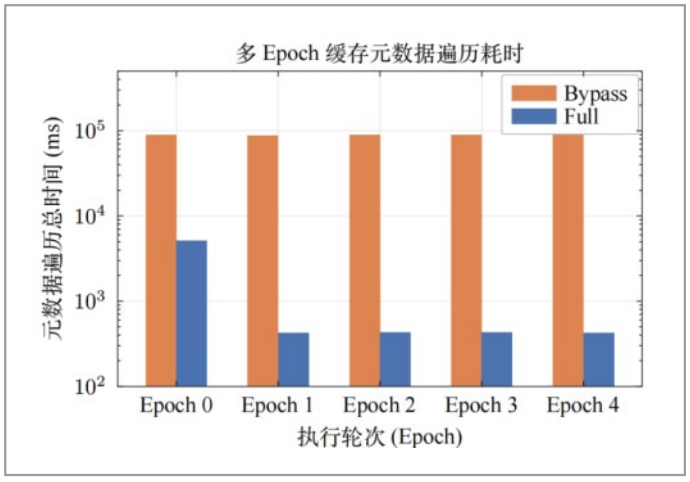


图 11　多 epoch 缓存元数据遍历耗时对比

具体来说，对于 ReadDir 操作，Full 模式在 5K 和 50K 文件规模下分别获得约 7.8 倍和 7.6 倍加速；对于 GetAttr 操作，5K 文件规模下由 8 598.29 ms 降至 9.39 ms，50K 文件规模下由 87 508.40 ms 降至 4 006.30 ms。上述结果说明，批量聚合显著降低了细粒度元数据访问开销，结果如图 12 所示。

批量聚合使单次请求携带更多目录，其潜在代价是增大单请求时延、可能推迟首批训练样本就绪；为量化该开销，本文以聚合粒度 β 为变量进行了端到端测量（表 6、图 13）。请求级对照显示：单次请求携带的目录数由 1 增至 50 时，请求端

表 6　批量聚合粒度 β 对训练冷启动与首批样本就绪的影响

网络条件	β	目录遍历/s	首批就绪(含遍历)/s
默认链路(80 Mbit/s、RTT≈5 ms)	1	5.7	6.3
默认链路(80 Mbit/s、RTT≈5 ms)	10	1.4	1.9
默认链路(80 Mbit/s、RTT≈5 ms)	50	1.1	1.7
默认链路(80 Mbit/s、RTT≈5 ms)	200	1.1	1.6
云边链路(80 Mbit/s、RTT≈50 ms)	1	56.5	58.4
云边链路(80 Mbit/s、RTT≈50 ms)	10	7.7	9.5
云边链路(80 Mbit/s、RTT≈50 ms)	50	4.6	6.5
云边链路(80 Mbit/s、RTT≈50 ms)	200	4.6	6.5

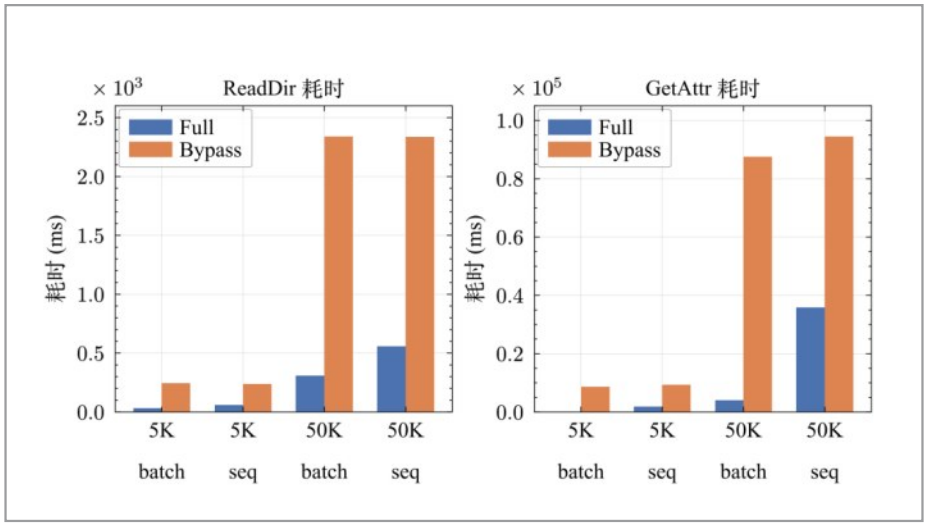


图 12 批量预取与逐条请求耗时对比

到端时延由 111 ms 增至 482 ms（云边链路 RTT≈50 ms；默认链路 RTT≈5 ms 下由 10.7 ms 增至 68.0 ms），增量约为一到两个往返时延量级；其中聚合响应体量约 68 KB（1000 条目录项，约 68 B/条），带宽占比不足 2%，开销主要来自元数据服务端逐目录处理。而同一批 50 个目录若按细粒度逐个跨域读取，需约 50 次往返、合计约 5.6 s（云边），聚合节省的跨域往返远大于其引入的单请求增量。端到端测量证实批量聚合并未推迟首批训练数据就绪：β 由 1 增至 50 时，目录遍历耗时在云边链路由 56.5 s 降至 4.6 s（约 12 倍），默认链路由 5.7 s 降至 1.1 s（约 5 倍）；“训练启动→首批样本就绪（含目录遍历）”时延由 58.4 s 降至 6.5 s（云边）、由 6.3 s 降至 1.7 s（默认）；仅数据部分的首批就绪时延保持约 1.9 s/0.6 s（云边/默认链路）与 β 基本无关，即聚合并未将“多读元数据”的开销传导至首批数据读取，而是通过压缩遍历往返整体提前了样本就绪。此外，该开销仅发生在冷启动一次性遍历阶段，不会在训练循环内累积：Epoch-0/Epoch-1 时长与训练吞吐不随 β 变化（云边约 237.5 s 与 190~199 s，默认约 84.6 s 与 43~44 s）。综上，批量预取以约 0.37 s（云边）的单请求增量换取整库目录遍历约 51.9 s 的缩减，对首批训练数据就绪时间而言是净收益；该机制与数据预热、缓存命中共同构成端到端实验（图 7~图 10）中 CEDFS-full 显著优于直连方案的基础，且不会以推迟首批训练数据就绪为代价。

检查点机制实验进一步分为保存阻塞、分级恢复和自动路由恢复 3 组。三组实验关注的指标不同：图 14 比较检查点保存对训练前台的阻塞及本地恢复时间，图 15 比较不同恢复策略下训练可恢复时间的缩短效果，图 16 则比较节点状态变化和丢包对完整远端恢复时间的影响[13，15，18，20]。

首先比较四种检查点保存方法。图 14 的横轴为 torch.save、PyTorch DCP、CEDFS-bypass 和 CEDFS-full。torch.save 为 PyTorch 原生本地保存，不创建远端副本；PyTorch DCP 为分布式检查点接口；CEDFS-bypass 关闭本地优先与异步复制，采用同步网络写入；CEDFS-

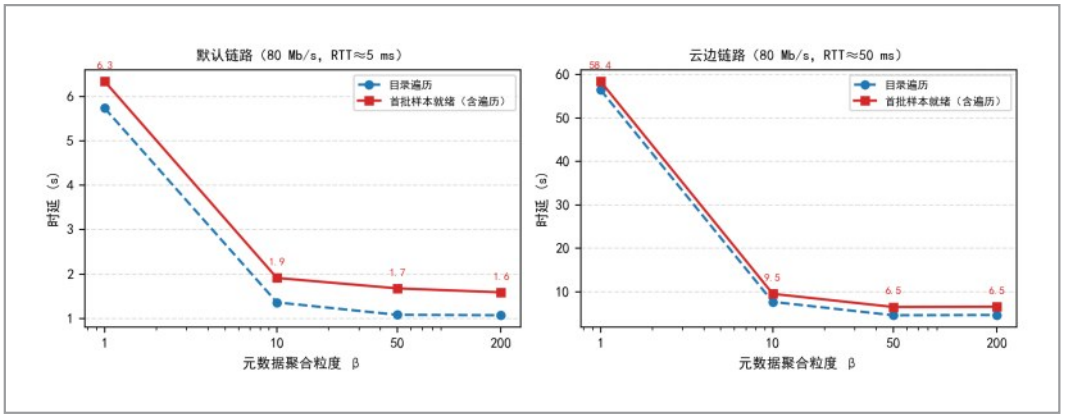


图 13　不同批量聚合粒度 β 下的目录遍历耗时与首批样本就绪时延（含遍历）

full 启用本地优先与异步复制。实验使用 175 MB ResNet-50 检查点，CEDFS SDK 并行写入线程数为 4。图中的“写入阻塞时间”为该组实验记录的保存阶段阻塞时间，“恢复时间 RTO”为本地副本可用时的加载恢复耗时。这里的“本地”指执行保存操作的客户端所在节点，不能将其等同于远端边侧存储。

结果如图 14 所示，CEDFS-full 的写入阻塞时间为 0.105 s，与 torch.save 的 0.092 s 接近，低于 PyTorch DCP 的 0.589 s 和 CEDFS-bypass 的 13.375 s。该结果用于比较本地优先与同步网络写入两类实验配置下的保存开销。0.105 s 对应本地优先实验中的保存阻塞指标，不等于全部云边副本的创建总时间，也不表示完整检查点从云侧同步传输至边侧的耗时。异步复制仍产生网络和存储开销，实际保存调用的等待范围由所采用的保存路径和完成条件决定。图 14 中四种方法的恢复时间均约为 0.03 s，反映本地副本完好可用时的恢复开销。

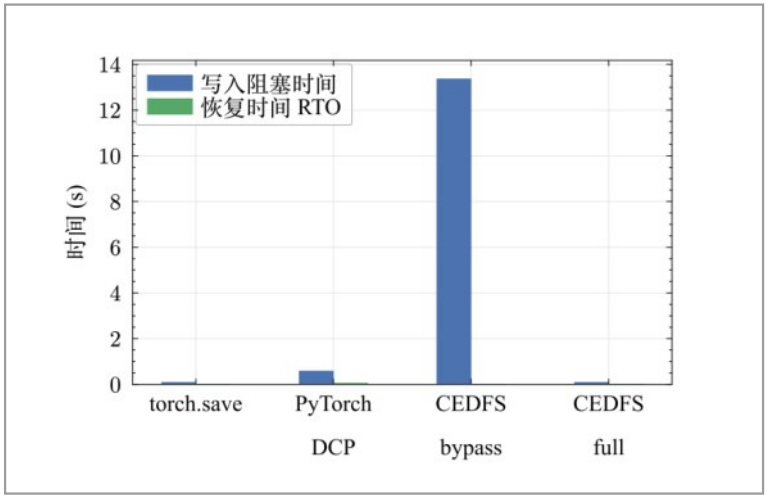


图 14　四种检查点保存方法的写入阻塞时间与恢复时间对比

随后比较分级恢复策略。这里的评价指标是“训练可恢复时间”，即恢复到训练能够继续执行所需的关键状态加载时间，而不是全部检查点数据都恢复完成的时间。恢复实验使用 239 MB 检查点，其中 Critical/High 级关键路径数据约 94 MB，其余约 145 MB 可在训练恢复后惰性加载。图 15 中，“均匀下载”表示完整读取 239 MB 后再恢复训练；“分级恢复”表示优先读取 Critical/High 级数据后恢复训练；“仅关键层”只读取 Critical 级数据，主要作为理论下界参照，不作为完整训练恢复策略。图 15(a) 中虚线 y=x 表示与均匀下载耗时相同，点越低说明恢复越快；图 15(b) 给出相对于均匀下载的缩短比例。

结果表明，在 RTT=5 ms 时，均匀下载耗时 34.61 s，分级恢复为 13.35 s；在 RTT=200 ms 时，均匀下载增至 710.28 s，分级恢复为 284.03 s。分级恢复在所有 RTT 条件下均将训练可恢复时间缩短 59.6%~61.4%。该比例与可延后恢复数据约占总检查点 60.7% 基本一致，说明恢复收益主要来自关键路径数据量裁剪，而不是某个特定 RTT 条件下的偶然优化。

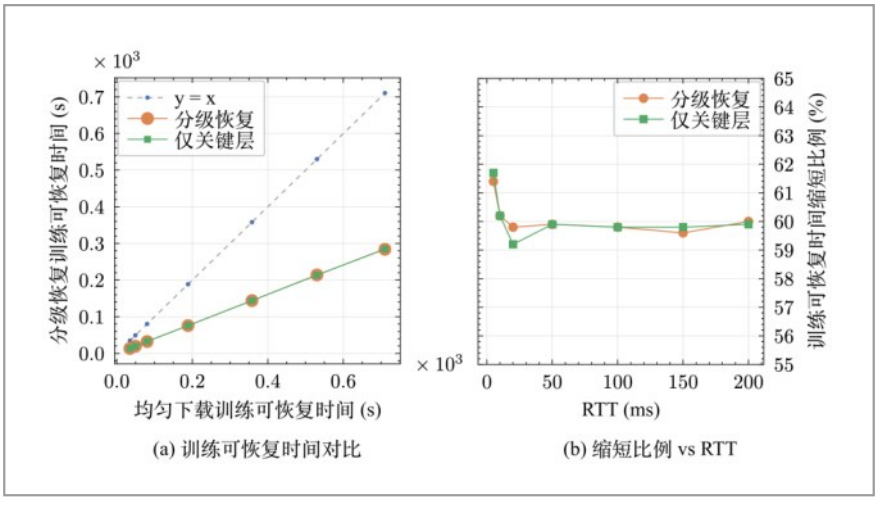


图 15 训练可恢复时间对比与缩短比例

最后评估自动路由恢复机制的稳健性。该实验仍使用 239 MB 检查点，但指标改为完整远端恢复时间 RTO，用于观察副本源异常时恢复过程是否被阻断。图 16 的横轴为首选副本源 Edge-A 的 3 种状态：正常（RTT=5 ms）、恶化（RTT=80 ms）和失效（节点不可达）；图例为 0%、1% 和 5% 三种丢包率；纵轴为恢复时间 RTO。检查点副本同时分布在 Edge-A、Edge-B（RTT=30 ms）和 Cloud（RTT=70 ms）上，每组配置重复 3 次取均值。

如图 16 所示，在相同丢包率下，Edge-A 正常、恶化和失效三组柱子的高度基本一致：0% 丢包下均约 257 s，1% 丢包下为 340~344 s，5% 丢包下为 659~665 s。这说明当首选副本源变慢或不可达时，CEDFS 能够自动切换到 Edge-B 或 Cloud 副本，避免单个副本源异常直接阻断恢复。另一方面，柱子随丢包率升高显著变高，说明该实验中的主要耗时来自底

层重传和拥塞退避；因此图 16 证明的是自动路由的稳定性，而图 15 证明的是分级恢复对训练可恢复时间的缩短效果。

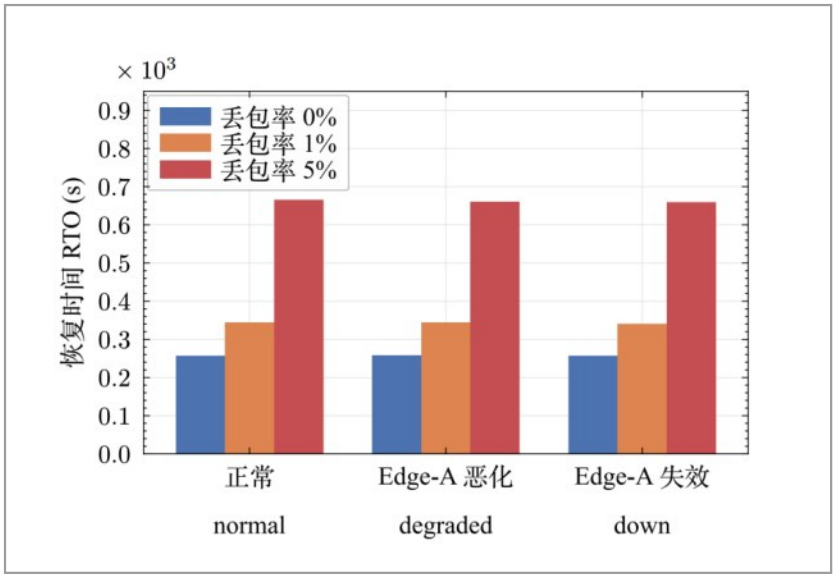


图 16　不同节点状态与丢包率下的检查点恢复时间

总体而言，三节点物理弱网环境证明了 CEDFS 对训练主流程的端到端收益，十节点云上可控环境进一步解释了该收益在元数据访问、检查点写入阻塞、分级恢复和自动路由恢复路径上的来源。两类实验设置分别服务于不同验证目标，所有结论均在同一环境内部与对应基线方案比较得到。

4　结束语

云边端跨域分布式训练对统一、低阻塞、可恢复的存储支撑提出了更高要求。本文设计并实现了面向云边端跨域训练的存储加速系统 CEDFS，以统一命名空间为基础，将冷启动元数据访问、训练期样本供给和检查点容错恢复纳入同一存储框架进行协同优化。

实验结果表明，CEDFS 能够在弱带宽和大时延环境下稳定提升训练吞吐，并显著降低元数据遍历、检查点写入阻塞和训练可恢复时间。后续将结合更细粒度的链路状态与在线负载信息，自适应调整批量大小、预热窗口和缓存配额，并面向更多训练任务细化检查点状态分类与副本策略。

参考文献：

[1] LIU M, PAN L, LIU S. Collaborative Storage for Tiered Cloud and Edge: A Perspective of Optimizing Cost and La - tency[J]. IEEE Transactions on Mobile Computing, 2024, 23(12): 10885-10902. DOI:10.1109/TMC.2024.3384264.

[2] CHEN J, WANG Y, YE M, et al. A Secure Cloud-Edge Collaborative Fault-Tolerant Storage Scheme and Its Data

Writing Optimization[J]. IEEE Access, 2023, 11: 66506-66521. DOI: 10.1109/ACCESS.2023.3291452.

[3] ROY S, SARKAR S, SAHU A. Profit Maximization Using Collaborative Storage Management in Multi-Tier Edge-Cloud System[C]//Proceedings of the 30th IEEE International Conference on High Performance Computing, Data, and Analytics. Piscataway: IEEE, 2023: 309-318. DOI: 10.1109/HiPC58850.2023.00048.

[4] DE LIMA MELO V D, THIESSEN M, PANAS A, et al. PathFS: A File System for the Hierarchical Edge[C]//Proceedings of the 7th International Workshop on Edge Systems, Analytics and Networking. New York: ACM, 2024: 55-60. DOI:10.1145/3642968.3654822.

[5] COPSTEIN R N, DOTTI F L. Distributed file system for an edge-based environment[Z/OL]. arXiv:2006.09182, 2020.

[6] WANG L, YE S, YANG B, et al. DIESEL: A Dataset-Based Distributed Storage and Caching System for Large-Scale Deep Learning Training[C]//49th International Conference on Parallel Processing (ICPP). New York: ACM, 2020: 20: 1-20: 11. DOI: 10.1145/3404397.3404472.

[7] SUN B, YU X, ZHANG C, et al. SOLAR: A highly optimized data loading framework for distributed training of CNN-based scientific surrogates[Z/OL]. arXiv: 2211.00224, 2022.

[8] KHAN R I S, YAZDANI A H, FU Y, et al. SHADE: Enable Fundamental Cacheability for Distributed Deep Learning Training[C]//Proceedings of the 21st USENIX Conference on File and Storage Technologies (FAST). Berkeley: USENIX Association, 2023: 135-152.

[9] LIU J, NICOLAE B, LI D. Lobster: Load Balance-Aware I/O for Distributed DNN Training[C]//Proceedings of the 51st International Conference on Parallel Processing. New York: ACM, 2022: 1-11.

[10] GU R, ZHANG K, XU Z, et al. Fluid: Dataset Abstraction and Elastic Acceleration for Cloud-native Deep Learning Training Jobs[C]//2022 IEEE 38th International Conference on Data Engineering (ICDE). Piscataway: IEEE, 2022: 2182-2195.

[11] GARG S, ZHANG J, PITCHUMANI R, et al. CrossPrefetch: Accelerating I/O Prefetching for Modern Storage[C]//Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1. New York: ACM, 2024: 102-116.

[12] DRYDEN N, B HRINGER R, BEN-NUN T, et al. Clairvoyant Prefetching for Distributed Machine Learning I/O[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC). New York: ACM, 2021: 1-15. DOI: 10.1145/3458817.3476181.

[13] WANG Z, JIA Z, ZHENG S. GEMINI: Fast Failure Recovery in Distributed Training with In-Memory Checkpoints [C]//Proceedings of the 29th Symposium on Operating Systems Principles. New York: ACM, 2023: 364-381.

[14] WANG G, RUWASE O, XIE B, et al. FastPersist: Accelerating Model Checkpointing in Deep Learning[Z/OL]. arXiv: 2406.13768, 2024.

[15] MAURYA A, UNDERWOOD R, RAFIQUE M M, et al. DataStates-LLM: Lazy Asynchronous Checkpointing for Large Language Models[C]//Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed

Computing. New York: ACM, 2024: 227–239.

[16] WAN B, HAN M, SHENG Y, et al. ByteCheckpoint: A unified checkpointing system for large foundation model development[C]//Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2025: 559–578.

[17] WANG Z, ZHU W, ZHANG Z, et al. Amber: Towards Fast and Space-Efficient Incremental Checkpointing in Large Language Model Training[C]//Proceedings of the 54th International Conference on Parallel Processing. New York: ACM, 2025: 678–688. DOI: 10.1145/3754598.3754606.

[18] MOHAN J, PHANISHAYEE A, RANIWALA A, et al. CheckFreq: Frequent, Fine-Grained DNN Checkpointing [C]//Proceedings of the 19th USENIX Conference on File and Storage Technologies (FAST). Berkeley: USENIX Association, 2021: 203–216.

[19] ZHANG K, CHEN Y, HU Z, et al. GoCkpt: Gradient-Assisted Multi-Step Overlapped Checkpointing for Efficient LLM Training[Z/OL]. arXiv: 2511.07035, 2025.

[20] HUANG Z, NIE H, JIA H, et al. FlowCheck: Decoupling Checkpointing and Training of Large-Scale Models[C]//Proceedings of the 20th European Conference on Computer Systems. New York: ACM, 2025: 1334–1349. DOI: 10.1145/3689031.3696088.

[21] LI W, CHEN X, SHU H, et al. ExCP: Extreme LLM Checkpoint Compression via Weight-Momentum Joint Shrinking[C]//Proceedings of the 41st International Conference on Machine Learning. PMLR, 2024: 27575–27588.

[22] AGRAWAL A, REDDY S, BHATTAMISHRA S, et al. Inshrinkerator: Compressing Deep Learning Training Checkpoints via Dynamic Quantization[C]//Proceedings of the ACM Symposium on Cloud Computing. New York: ACM, 2024: 1012–1031.

[23] JIN H, WU D, ZHANG S. Design of a Quantization-Based DNN Delta Compression Framework for Model Snapshots and Federated Learning[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(3): 923–937.

[24] PAWLOWSKI B, JUSZCZAK C, STAUBACH P, et al. NFS version 3: design and implementation[C]//Proceedings of the USENIX Summer Technical Conference. Berkeley: USENIX Association, 1994: 137–152.

[25] WEIL S A, BRANDT S A, MILLER E L, et al. Ceph: a scalable, high-performance distributed file system[C]//Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2006: 307–320.

[26] REN K, ZHENG Q, PATIL S, et al. IndexFS: Scaling file system metadata performance with stateless caching and bulk insertion[C]//SC14: International Conference for High Performance Computing, Networking, Storage and Analysis. Piscataway: IEEE, 2014: 237–248. DOI:10.1109/SC.2014.25.

[27] NIAZI S, ISMAIL M, HARIDI S, et al. HopsFS: Scaling Hierarchical File System Metadata Using NewSQL Databases [C]//Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST). Berkeley: USENIX Association, 2017: 89–104.

[28] GUO H, LU Y, LV W, et al. SingularFS: A Billion-Scale Distributed File System

Using a Single Metadata Server[C]// Proceedings of the USENIX Annual Technical Conference (ATC). Berkeley: USENIX Association, 2023: 915–928.

[29] LI S, LU Y, SHU J, et al. LocoFS: A loosely-coupled metadata service for distributed file systems[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2017: 1–12. DOI: 10.1145/3126908.3126928.

[30] LV W, LU Y, ZHANG Y. InfiniFS: An Efficient Metadata Service for Large-Scale Distributed Filesystems[C]//Proceedings of the 20th USENIX Conference on File and Storage Technologies (FAST). Berkeley: USENIX Association, 2022: 313–328.

[31] WANG Y, WU Y, LI C, et al. CFS: Scaling Metadata Service for Distributed File System via Pruned Scope of Critical Sections[C]//Proceedings of the 18th European Conference on Computer Systems. New York: ACM, 2023: 331–346. DOI:10.1145/3552326.3587443.

[32] 舒继武, 陈游旻, 王庆, 等. 面向解聚数据中心的存储系统进展[J]. 中国科学: 信息科学, 2023, 53(8): 1503–1545. DOI: 10.1360/ssi–2023–0034.
SHU J W, CHEN Y M, WANG Q, et al. Progress on storage systems for disaggregated data centers[J]. Scientia Sinica Informationis, 2023, 53(8): 1503–1545. DOI:10.1360/ssi–2023–0034.

[33] MORRIS J H, SATYANARAYANAN M, CONNER M H, et al. Andrew: a distributed personal computing environment[J]. Communications of the ACM, 1986, 29(3): 184–201.

[34] NELSON M N, WELCH B B, OUSTERHOUT J K. Caching in the sprite network file system[J]. ACM Transactions on Computer Systems (TOCS), 1988, 6(1): 134–154.

[35] TATEBE O, HIRAGA K, SODA N. Gfarm grid file system[J]. New Generation Computing, 2010, 28(3): 257–275.

[36] WANG Y, LI C, SHAO X, et al. Lunule: An Agile and Judicious Metadata Load Balancer for CephFS[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2021: 1–16.

[37] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop distributed file system [C]//2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. Piscataway: IEEE, 2010: 1–10. DOI: 10.1109/MSST.2010.5496972.

[38] LI H, GHODSI A, ZAHARIA M, et al. Tachyon: Reliable, memory speed storage for cluster computing frameworks [C]//Proceedings of the ACM Symposium on Cloud Computing. New York: ACM, 2014: 1–15. DOI: 10.1145/2670979.2670985.

[39] 金国栋, 卞昊穹, 陈跃国, 等. HDFS存储和优化技术研究综述[J]. 软件学报, 2020, 31(1): 137–161.
JIN G D, BIAN H Q, CHEN Y G, et al. Survey on storage and optimization techniques of HDFS[J]. Journal of Software, 2020, 31(1): 137–161.

[40] ZHANG Y, LI G, LU K, et al. PhatKV: Towards an efficient metadata engine for KV-based file systems on modern SSD[C]//Proceedings of the 38th International Conference on Massive Storage Systems and Technology. Santa Clara: MSST, 2024.

[41] HE Q, TAN S, CHEN F, et al. EDIndex: Enabling Fast Data Queries in Edge Storage Systems[C]//Proceedings of the 46th International ACM SIGIR Confer-

ence on Research and Development in Information Retrieval. New York: ACM, 2023: 675-685.

[42] XIE J, QIAN C, GUO D, et al. Efficient Indexing Mechanism for Unstructured Data Sharing Systems in Edge Computing[C]//IEEE International Conference on Computer Communications (INFOCOM). Piscataway: IEEE, 2019: 820-828.

[43] KUMAR A V, SIVATHANU M. Quiver: An Informed Storage Cache for Deep Learning[C]//Proceedings of the 18th USENIX Conference on File and Storage Technologies (FAST). Berkeley: USENIX Association, 2020: 283-296.

[44] 明龙, 胡建龙, 张心译, 等. 基于动态权衡模型与SDAPC框架的Hadoop生态系统优化：跨层级存储与计算协同及可信调度策略[J]. 大数据, 2026, 12(4): 97-116.
MING Long, HU Jianlong, ZHANG Xinyi, et al. Optimization of the Hadoop ecosystem based on the dynamic trade-off model and SDAPC framework: cross-layer storage and computation synergy and trustworthy scheduling strategies [J]. Big Data Research, 2026, 12(4): 97-116.

[45] 王锦权, 刘旭昭, 廖晓坚, 等. SpanTrain：基于云边端异构设备的跨域分布式模型训练系统[J]. 大数据, 2025, 11(3): 17-32.
WANG Jinquan, LIU Xuzhao, LIAO Xiaojian, et al. SpanTrain: a cross-domain distributed model training system for cloud-edge-end heterogeneous devices[J]. Big Data Research, 2025, 11(3): 17-32.

作者简介



刘旭昭（2000– ），男，北京航空航天大学博士生，主要研究方向为存储系统、分布式训练与并行计算。



谭升阳（2000– ），男，北京航空航天大学硕士生，主要研究方向为存储系统。



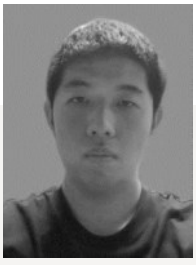
廖晓坚（1994– ），男，博士，北京航空航天大学计算机学院副教授，主要研究方向为内存/存储系统、分布式系统与智能计算系统。



霍志胜（1983– ）男，博士，北京航空航天大学计算机学院助理研究员，主要研究方向为大数据存储、分布式/并行存储系统的框架研究及性能优化、大数据环境中重复数据删除。



王锦权（1998– ）男，北京航空航天大学博士生，主要研究方向为分布式训练与并行计算。



陈奕铭（2002– ）男，北京航空航天大学硕士生，主要研究方向为存储系统与智能计算系统。

录用日期: XXXX-XX-XX

通信作者: 廖晓坚,liaoxj@buaa.edu.cn;霍志胜,huozhisheng1122@126.com;

基金项目: 国家重点研发计划资助项目(No. 2023YFB4503100);国家自然科学基金资助项目(No. 62572022);中央高校基本科研业务费专项资金。作者廖晓坚另获北京市科学技术协会青年人才托举工程资助。

Foundation Items: The National Key Research and Development Program of China (No. 2023YFB4503100); the National Natural Science Foundation of China (No. 62572022); the Fundamental Research Funds for the Central Universities. Xiaojian Liao is additionally supported by the Young Elite Scientist Sponsorship Program of the Beijing Association for Science and Technology.