计 算 机 研 究 与 发 展
Journal of Computer Research and Development

https://crad.ict.ac.cn
63(3): 627−639, 2026

HiTrain: 面向大模型训练的异构内存卸载与 I/O 优化

汤志航[1,2] 兰 颢[3] 刘政国[4] 汪 庆[3] 赵来平[1] 李克秋[1]

[1](天津大学智能与计算学部 天津 300350)

[2](之江实验室高效能计算设施研究中心 杭州 311121)

[3](清华大学计算机科学与技术系 北京 100084)

[4](厦门大学信息与通信工程系 福建厦门 361005)

(tangzhihang@tju.edu.cn)

HiTrain: Heterogeneous Memory Offloading and I/O Optimization for Large Language Model Training

Tang Zhihang[1,2], Lan Hao[3], Liu Zhengguo[4], Wang Qing[3], Zhao Laiping[1], and Li Keqiu[1]

[1](*College of Intelligence and Computing*, *Tianjin University*, *Tianjin* 300350)

[2](*Research Center for High Efficiency Computing Infrastructure*, *Zhejiang Lab*, *Hangzhou* 311121)

[3](*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)

[4](*Department of Information and Communication Engineering*, *Xiamen University*, *Xiamen*, *Fujian* 361005)

Abstract With the continuous growth of the parameter scale of large language models (LLMs), fine-tuning models with hundreds of billions of parameters pose extremely high requirements for computing and storage resources. Traditional distributed training schemes often rely on a large number of high-end graphics processing units (GPUs) and high-speed interconnect networks, resulting in extremely high training costs. Although the existing single-GPU training schemes relieve the pressure on GPU memory through tensor offloading, they still face problems such as low I/O transmission efficiency and insufficient device utilization. Traditional kernel-space I/O operations introduce frequent system calls and context switching during large-scale tensor transfers, becoming a key bottleneck that restricts performance. Meanwhile, the optimizer computations fail to fully leverage the parallel capabilities of multi-core CPUs, making it difficult to effectively overlap with GPU computations, further limiting system performance. In response to the above problems, we propose HiTrain, a heterogeneous memory offloading and I/O optimization scheme for LLM training, with a focus on the design and implementation of two key technologies: First, it constructs a high-performance tensor storage module based on the storage performance development kit (SPDK), which manages tensor data in user space, thereby avoiding the overhead of the kernel I/O stack and improving the concurrency and throughput of tensor offloading. Second, it designs and implements an asynchronous-optimizer based storage-compute pipeline scheduling module, which reorders optimizer execution to reduce GPU idle time and thus enhance overall training efficiency. The experimental results show that on the server equipped with a single GPU and a non-volatile memory express solid state drive (NVMe SSD), the scheme proposed can make full use of the computing and storage resources in the system, improve the efficiency of tensor unloading and loading in the process of model training by 32.7%, and increase the overall training throughput to 1.49 times of the existing scheme, offering a practical and cost-effective path for LLM training.

收稿日期: 2025-06-17; **修回日期**: 2025-10-16

基金项目: 国家重点研发计划项目(2024YFB4505200)

This work was supported by the National Key Research and Development Program of China (2024YFB4505200).

通信作者: 兰颢(lanhao@tsinghua.edu.cn)

Key words large model training; heterogeneous memory; tensor offload; user-space high-performance I/O; storage-computing pipeline

摘 要 随着大语言模型(large language models, LLMs)(以下简称“大模型”)参数规模的持续增长,微调百亿级参数大模型对计算和存储资源提出了极高要求。传统分布式训练方案通常依赖大量高端GPU和高速互联网络,训练成本极为昂贵。现有单GPU训练方案虽通过张量卸载缓解显存压力,但仍然面临I/O传输效率低和设备利用率不足等问题。传统内核态I/O操作在大规模张量迁移中引入频繁的系统调用和上下文切换,成为制约性能的关键瓶颈;同时,优化器计算无法充分发挥多核CPU的并行能力,难以实现与GPU计算的有效重叠,进一步限制了系统性能。针对上述问题,提出了一种面向大模型训练的异构内存卸载与I/O优化方案HiTrain。首先构建了基于存储性能开发工具包(storage performance development kit, SPDK)的高性能张量存储模块,通过在用户态管理张量数据,避免了内核I/O栈开销,从而提高张量卸载的并发性与吞吐率;其次,设计并实现了基于异步优化器的存储-计算流水线调度模块,通过对优化器的执行进行优化重排来减少GPU等待时间,提高整体训练效率。实验结果表明,在配备单张GPU和非易失性存储器快速固态硬盘(non-volatile memory express solid state drive, NVMe SSD)的服务器上,所提出的方案能够充分利用系统中的存算资源,使得模型训练过程中张量卸载与加载效率提升32.7%,整体训练吞吐提升至现有方案的1.49倍,为低成本大模型训练提供了切实可行的技术路径。

关键词 大模型训练;异构内存;张量卸载;用户态高性能I/O;存储-计算流水线

中图法分类号 TP333; TP391.9

DOI: 10.7544/issn1000-1239.202550478 **CSTR:** 32373.14.issn1000-1239.202550478

近年来,随着Transformer[1]等模型架构的不断演进,大语言模型(large language models, LLMs)(以下简称“大模型”)在智能问答[2]、代码生成[3]和多模态学习[4]等多个领域不断取得突破性进展。伴随模型能力的不断增强,其参数规模也呈现出指数级增长趋势,从最初的百万级迅速扩展至当前的千亿至万亿级。

模型规模的激增带来性能跃升的同时,对计算与存储资源的需求也急剧增加,尤其是对显存容量提出了前所未有的挑战[5]。以微调一个100亿参数的大模型为例,其在训练期间所需存储的各类张量数据总量的峰值高达2.6 TB[6],远超单张GPU所能提供的显存容量上限。

如图1所示,尽管硬件性能持续提升,但大模型参数规模的增长速度仍远超GPU显存容量的扩展速度,且二者之间的差距仍在不断扩大。即便是当前最先进的高端专用AI加速器(如配备288 GB显存的AMD MI350系列),其显存容量仍然显著低于当前大模型训练所需的内存规模。这一趋势表明,大模型对存算资源的需求与硬件发展之间的矛盾正日益加剧,已成为制约其训练效率与可扩展性的关键瓶颈。

随着通用预训练模型在各类任务中的广泛应用,

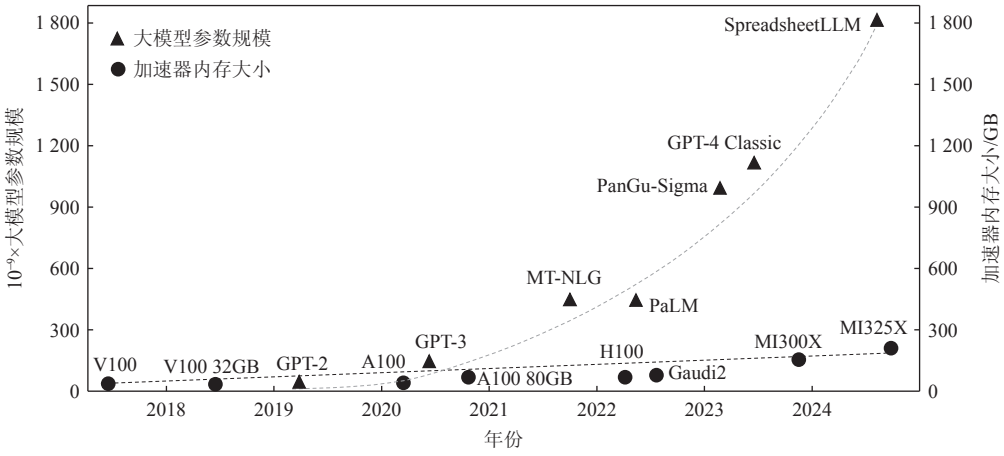


Fig. 1 Comparison of model size and growth speed of accelerator memory

图1 模型规模和加速器内存增长速度对比

越来越多的普通研究机构和开发者希望通过模型微调将大模型适配于具体的下游任务[7-9]。与预训练阶段主要依赖大规模计算集群不同，微调训练任务通常在资源有限的单机或中小型服务器上进行，难以满足当前主流大模型训练方案对显存资源的极高要求。如 DeepSpeed[10]，Megatron-LM[11] 等分布式训练框架虽然通过模型并行[12]、流水线并行[13] 以及专家并行[14] 等策略有效缓解了显存压力，并支持百亿至千亿级参数模型的训练[15]，但其通常依赖于大量高端 GPU 和高速互联网络（如 InfiniBand[16]，NVLink[17]），部署成本极为昂贵。例如，微调一个 1000 亿参数的大模型通常至少需要 32 张英伟达 A100 GPU，总硬件投入超过 45 万美元，普通研究机构和中小开发者难以承受。

为降低大模型训练门槛，近年来研究者们提出了内存卸载技术，尝试将训练过程中的模型状态（即优化器状态、梯度、模型参数）和激活值等核心张量从 GPU 显存卸载至 CPU 内存或固态硬盘（solid state drive，SSD）[18]，从而在有限资源条件下支持更大规模参数的大模型训练。典型方案如 ZeRO[19]，ZeRO-Offload[20]，ZeRO-Infinity[21]，LoHan[22] 等，虽然在突破显存容量限制方面取得了显著进展，拓展了大模型训练在低成本硬件上的可行性，但仍存在诸多不足：一方面，内存卸载带来的训练过程中显存与 SSD 之间频繁的张量迁移引发大量的高速串行计算机扩展总线 PCIe（peripheral component interconnect express）传输开销，而现有卸载方案在 SSD 数据读写上仍依赖于传统内核 I/O 栈，在高频读写场景下，系统调用、上下文切换和中断处理等软件开销成为主要性能瓶颈。另一方面，大模型训练在反向传播过程中需要将模型参数和优化器状态从 SSD 上加载至显存或从显存卸载回 SSD，现有方案的存储-计算流水线调度策略难以有效掩盖加载/卸载过程带来的 I/O 开销，导致 GPU 资源存在空闲状态，整体利用率偏低。因此，如何优化 I/O 路径、提升带宽利用率并降低 GPU 空闲时间是当前低成本大模型训练系统性能优化亟待解决的关键挑战。

针对上述问题，本文基于“高性能张量 I/O”与“异步优化器流水线”的核心思想，提出一种面向大模型训练的异构内存卸载与 I/O 优化方案 HiTrain，以缓解大模型训练过程中的存储与计算瓶颈，提升大模型在普通单 GPU 或中小型服务器上的训练效率。本文主要贡献包括：

1）构建了基于存储性能开发工具包（storage performance development kit，SPDK[23]）的高性能张量存储模块，并将其集成至主流大模型训练框架 DeepSpeed 中，以充分发挥 SSD 设备的硬件性能，降低张量卸载的 I/O 路径延迟，提高存储带宽利用效率。

2）设计并实现了基于异步优化器的存储-计算流水线调度模块，通过重构训练过程中优化器的执行时序与内存调度策略，均衡前向传播与反向传播过程中的存储与计算负载，充分减少 GPU 等待时间，提升系统整体资源利用率与训练吞吐。

3）在多个典型的大模型微调训练任务上对 HiTrain 进行了性能评估，实验结果表明，在配备单张消费级 GPU 与少量非易失性存储器快速接口（non-volatile memory express，NVMe）SSD 设备上，HiTrain 的训练效率与硬件资源利用率显著优于现有方案，张量卸载与加载效率提升了 32.7%，整体训练吞吐提升至现有方案的 1.49 倍。

1 相关工作

为突破大模型训练面临的显存瓶颈与资源限制，提升模型训练的可扩展性与资源利用率，近年来国内外诸多研究团队围绕内存卸载技术开展了广泛研究。早期相关工作通过将模型状态卸载到 CPU 内存以缓解显存压力，而随着大模型参数规模的不断扩展，仅依靠 CPU 内存的卸载方式无法满足训练过程中庞大的张量卸载需求，大模型训练的内存卸载方式逐步演变至将模型状态和激活值卸载至 CPU 内存和 SSD 等异构内存设备，并结合多种流水线调度与计算重叠优化策略，显著提升了大模型训练效率和资源利用率。本节详细介绍相关工作。

微软公司于 2021 年基于 DeepSpeed 框架提出 ZeRO-Offload[20]，在训练过程中将优化器状态和梯度从 GPU 卸载到 CPU 内存，从而显著缓解 GPU 显存压力。随后推出的 ZeRO-Infinity[21] 在 ZeRO-Offload 基础上进一步引入张量分级卸载机制，将模型状态扩展卸载至 SSD，并结合重计算策略将优化器计算迁移至 CPU 执行，同时引入了 GDS（GPUDirect storage）[24] 技术以实现 GPU 与 SSD 间的直接通信，大幅提升了数据传输效率。该方案在单个 NVIDA DGX-2 节点上成功实现了万亿参数模型的微调。然而，ZeRO-Infinity 也存在优化器与反向传播无法并行导致 GPU 利用率下降、重计算策略引入额外计算开销以及激活值仅支持卸载至 CPU 内存从而限制了可训练模型上限等局限性。

韩国首尔大学和三星电子团队于2021年提出FlashNeuron[18]，其通过冷热数据分离机制，将低频访问的优化器状态与部分模型参数卸载至Flash存储，有效缓解了显存压力，结合自适应缓存算法减少了冗余数据加载，同时引入了选择性卸载策略平衡显存占用与计算开销。然而该方案受限于Flash存储有限的读写带宽，随着模型规模逐渐增大卸载的数据量逐渐增多，I/O带宽逐渐成为性能瓶颈。同时该方案依赖于特定的硬件配置，如需支持GPU与Flash间的直连通信，限制了其在通用系统上的部署与推广。

在提高系统适配性方面，HPC-AI Tech团队在其2023年推出的Colossal-AI[15]系统中提出了更为通用的异构内存管理框架。其核心模块Gemini通过自动跟踪张量生命周期，将模型参数与块间激活值在GPU与CPU内存之间进行动态调度卸载，块内激活值则通过重计算维护，极大提升了系统可用性与训练效率。该框架具备良好的易用性与兼容性，能够集成至主流训练框架中使用，但其卸载路径主要面向CPU内存，未对SSD等更大容量设备提供深度优化支持，且优化器–计算重叠调度能力有限，影响了其在极限资源条件下的扩展性与效率表现。

关于激活值卸载到SSD方面，美国英伟达公司联合相关团队2023年提出SSDTrain[25]，通过精细管理激活值张量生命周期，将激活值从GPU显存异步卸载至NVMe SSD，并在反向传播阶段精确预取回显存，实现数据与计算的流水线重叠，显著降低了显存压力。在不引入额外重计算的前提下，有效支持更大模型规模和更高吞吐的训练流程。但同样其性能高度依赖于底层存储设备的带宽，主要面向支持GDS技术的高端GPU和高速NVMe SSD的服务器环境。

为进一步推进内存卸载粒度扩展，美国伊利诺伊大学香槟分校2023年提出G10[26]，该方案将激活值与模型状态统一卸载至CPU内存或SSD，避免重计算策略带来的额外计算开销，并在GPU上执行优化器计算，卸载激活值至SSD后在反向传播中加载回显存。然而由于需要在每轮迭代中频繁地将大规模张量通过PCIe通道传输回GPU，系统整体性能受到严重的I/O瓶颈限制。例如在训练130亿参数模型时，单轮迭代需传输的数据高达182 GB，传输时延远超GPU计算时间。同时，G10同样依赖GDS技术，无法兼容消费级GPU平台，限制了其部署的灵活性和通用性。

为了提高内存卸载过程中CPU与GPU之间数据的传输效率，加州大学默塞德分校等团队于2024年提出TECO[27]，其基于CXL(compute express link)技术优化大模型训练中的张量卸载效率。该方案通过细粒度缓存一致性策略和“脏字节聚合”机制，仅传输实际更新数据，显著减少PCIe带宽占用和降低通信开销。TECO具备良好的兼容性，可集成至现有框架如DeepSpeed，但由于其依赖于支持CXL的高端硬件，难以广泛适用于消费级GPU场景。

为提升卸载策略在通用硬件上的适用性，浙江大学和香港中文大学团队针对消费级硬件场景的训练限制2024年提出系统级优化方案LoHan[22]，旨在支持百亿参数级模型在单张GPU上的高效训练。其通过主动梯度卸载与子组优化器并行计算，提高GPU利用率，采用任务重叠机制，使优化器计算与I/O操作并发执行，提升带宽利用率，并提出流量感知的激活交换策略，在重计算与卸载之间动态权衡，以减少迭代延迟。然而其性能受限于SSD读写带宽，频繁的张量卸载与加载操作易形成训练瓶颈，且激活值交换过程中需额外存储中间激活张量，进一步加重I/O负担，限制了系统在高密度训练负载下的可扩展性。

综上所述，如表1所示，现有内存卸载和计算优化研究工作在提升大模型训练的可扩展性与资源适配性方面已取得了显著进展，部分方案实现了百亿至万亿参数模型的训练支持，同时也出现了如LoHan等面向消费级平台的内存卸载优化方案。然而仍面

Table 1 Comparison of Different Schemes on Memory Offloading

表1 不同内存卸载方案对比

方法	内存优化机制	优化器位置	重计算	性能瓶颈
FlashNeuron	模型状态→Flash 部分激活值→Flash	GPU	需要	Flash存储带宽有限。
ZeRO-Infinity	模型状态→SSD 激活值→CPU内存	CPU	需要	CPU-GPU计算非重叠导致GPU空闲；大量重计算导致PCIe带宽闲置； 激活值卸载到CPU内存产生额外卸载开销。
SSDTrain	激活值→SSD	GPU	不需要	性能高度依赖于底层存储设备的带宽。
G10	模型状态→SSD 激活值→SSD	GPU	不需要	反向传播时GPU需等待SSD传输优化器状态；与大量激活值导致GPU闲置。
LoHan	模型状态→SSD 部分激活值→CPU内存	CPU	需要	激活值卸载到CPU内存产生额外卸载开销； I/O操作的上下文切换开销大。

临3个关键挑战：1)卸载路径中的频繁张量迁移对I/O通道带宽提出极高要求，而多数现有方法采用内核态异步I/O接口(如libaio)处理NVMe SSD的数据访问，存在频繁的系统调用、上下文切换和中断处理开销，难以充分发挥NVMe SSD的高并发、高吞吐特性，在高频张量迁移场景下成为性能瓶颈；2)优化器计算与数据传输缺乏有效的流水线重叠调度方案，导致GPU空闲时间增加，计算资源利用率不高；3)部分方案通过GDS技术实现GPU与SSD间的直接通信，依赖特定高端硬件降低了方案的通用性与可部署性，此外GDS仅适用于GPU读取FP16精度参数的场景，且GPU与CPU同时发起的I/O操作会相互干扰，导致I/O时间延长，影响整体系统吞吐。针对上述问题，大模型训练内存卸载技术亟需在I/O卸载路径重构、计算–通信重叠调度机制以及通用系统适配性等方面进一步探索。

2 问题分析

本节围绕大模型训练过程中张量卸载所引发的I/O瓶颈以及现有流水线机制在计算与存储重叠方面的局限性，从张量卸载的存储需求、内核态I/O接口的访问开销以及计算与数据传输协同机制等维度出发，深入探讨当前大模型训练系统在高频张量迁移场景下存在的资源利用率低、训练吞吐受限等核心问题，以下进行详细分析。

1)张量卸载需求庞大，训练显存压力持续上升。在大模型训练迭代过程中，系统不仅需要存储模型参数，还需同时维护优化器状态、前向传播中产生的激活值以及反向传播中的梯度等多类张量。如表2所示，对于参数量为N的模型，优化器状态(OS_{32})与梯度(G_{16})的存储开销分别为$8N$和$2N$，约为模型参数(P_{16})存储开销的5~6倍，而激活值(A_{16})的存储开销则与批大小(Bs)、上下文长度(L)、词元维度(D)以及模型层数(Ln)呈线性关系，其总量约为上述四者乘积的2倍，随着大模型上下文窗口的持续扩展，训练过程中张量总存储开销急剧上升，进一步加剧了系统的内存压力。以OPT-13B模型为例，训练过程的张量总存储需求超过180 GB，然而普通服务器的CPU内存容量通常仅为64~128 GB，因此需要采用SSD等存储设备进行张量卸载。在这种条件下，I/O路径的传输性能成为训练效率的决定性因素，尤其在频繁交换张量与更新高并发梯度过程中，其性能瓶颈将直接影响训练吞吐与资源利用率。具体的训练参数如表2所示。

Table 2 Tensor Storage Overhead in LLM Training

表2 大模型训练过程中的张量存储开销

张量类型	符号	精度	存储开销/B	产生阶段
模型参数	P_{16}	FP16	$2N$	反向传播
优化器状态	OS_{32}	FP32	$8N$	反向传播
梯度	G_{16}	FP16	$2N$	反向传播
激活值	A_{16}	FP16	$2{\times}Bs{\times}L{\times}D{\times}Ln$	前向传播

2)内核态I/O接口开销高，难以释放SSD性能潜力。在基于张量卸载的大模型训练任务中(尤其是反向传播过程)，系统需要频繁地从SSD中加载张量或将张量卸载至SSD。如图2所示，在每轮反向传播过程中，系统首先需要将模型参数与对应优化器状态从SSD中加载至CPU内存；然后GPU加载参数并将计算的梯度卸载至CPU内存；最后CPU在执行完优化器更新后将新参数与优化器状态写回SSD。在此过程中，系统I/O接口的效率至关重要。

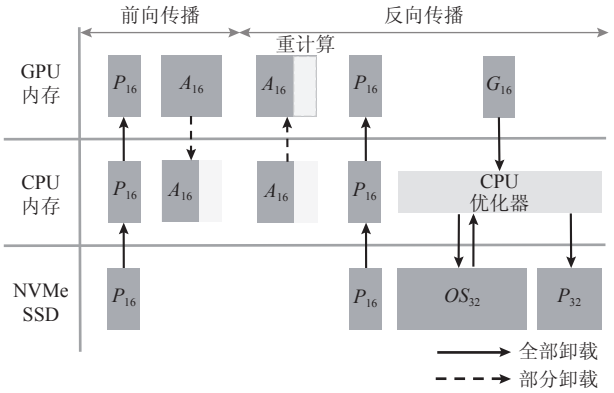


Fig. 2 Update steps of optimizer

图2 优化器更新步骤

尽管NVMe SSD具备较好的读写性能，然而现有张量卸载方案主要依赖libaio(asynchronous I/O library)[28-29]等Linux内核提供的传统内核态I/O接口访问SSD，需频繁进行用户态/内核态切换、中断处理及权限校验，I/O路径过程中软件开销高[30-32]，如图3所示，一个I/O请求发起后首先由用户空间进入内核空间，切换开销为0.5~2 μs[33]，读写请求在内核的软件层的时间开销为5~15 μs[34]，其中包含队列管理、请求调度、权限检查等任务。当设备完成I/O后会向主机发起中断，大约需要若干微秒[35]，在此之后应用程序才能得知I/O完成。从而导致系统对带宽的实际利用率远未达到NVMe SSD硬件上限，以英特尔傲腾(Optane)数据中心型NVMe SSD为例，理想情况下，libaio在读写大块张量时应能达到7 GBps和4.7 GBps。

然而在实际训练负载中的测试结果表明, libaio 在参数加载与卸载阶段的 SSD 平均读写带宽约为 5.2 GBps 和 3.6 GBps, 仅达到带宽上限的 75%, 在读写较小的张量时带宽甚至衰减至 1.2 GBps, 存在严重的带宽浪费, 未能充分发挥 NVMe 设备的高性能特性, 具有较大的可提升空间。

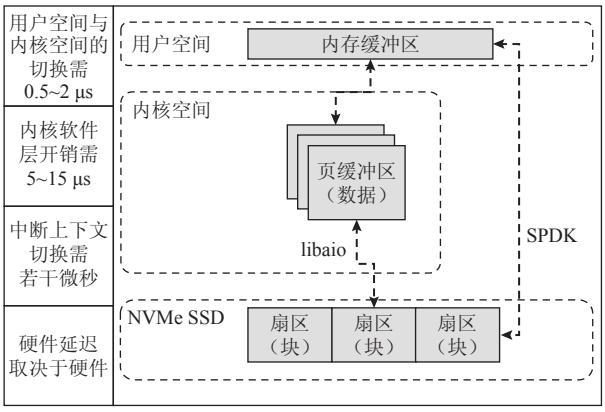


Fig. 3 Comparison of kernel-space and user-space I/O paths

图 3 内核态与用户态 I/O 路径对比

这是由于传统内核态 I/O 栈主要针对机械硬盘等慢速存储设备设计, 相对于慢速设备的读写延迟, 传统 I/O 路径中的软件开销可以忽略不计。然而, 随着 NVMe SSD 等新型高速存储介质的出现, libaio 内核态 I/O 路径的高额开销显著限制了硬件的发挥。

部分现有工作通过增加 NVMe SSD 数量来构建高性能 0 级磁盘阵列 RAID0(redundant arrays of independent disks 0), 以聚合外部存储带宽来提高张量卸载的效率。在理想条件下, RAID0 的带宽可近似等于所有设备带宽的总和。然而, 该方案不仅带来显著的经济成本和能耗开销, 在实际部署中亦存在物理限制。面向单卡或消费级设备的场景, 受限于主板可用的 PCIe 通道数量, 系统很难连接足够数量的高速 NVMe SSD 来构建大规模高性能 RAID0 阵列。

3)存储-计算流水线重叠有限, 难以掩盖显著的张量卸载开销。现有系统尝试通过存储-计算流水线重叠机制, 在 GPU, CPU, SSD 之间交错调度张量卸载与参数更新操作, 以缓解 I/O 延迟带来的系统阻塞。图 4 展示了该机制的执行流程, 其中 Comp^i 表示 GPU 或 CPU 对第 i 个参数子组进行计算, 其他符号含义如表 2 所示。具体而言, 对于第 i 个参数子组, 系统在 GPU 完成反向传播后, 将其梯度卸载至 CPU 内存, 并从 SSD 加载对应的参数与优化器状态到 CPU 执行更新, 同时并行调度第 i+1, i+2 子组的数据操作以实现跨组重叠。然而, 由于底层 I/O 吞吐受限, SSD 的读写延迟远高于计算操作, 即使流水线调度得当, 也难以完全掩盖卸载开销, GPU 在多个阶段仍处于等待状态, 整体利用率低。

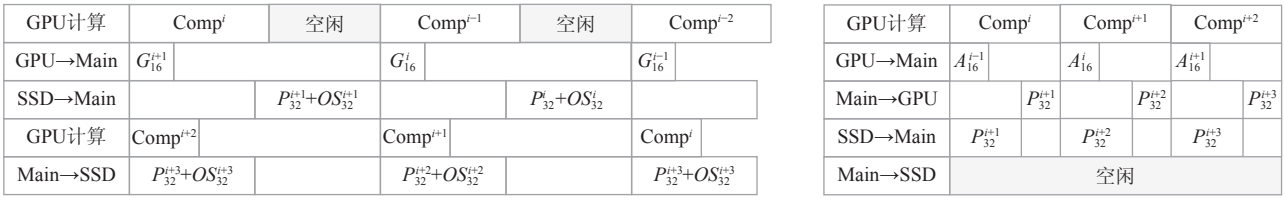
如图 4(b)所示, 与反向传播过程的存储瓶颈相反, LoHan 在前向传播过程中呈现出明显的计算瓶颈特征。这是由于前向传播过程 GPU 仅需要从 SSD 中拉取精度为 FP16 的模型参数, 张量加载的时间开销通常低于前向传播计算所需的时间。此外, 前向传播的计算时间随批大小与序列长度的增大而增加, 而所需加载的模型参数张量大小固定不变, 进一步加剧了存储带宽资源的浪费。初步分析的结果表明, 在 1 轮训练过程当中, LoHan 大约有 17.3% 的时间使 I/O 处于空闲状态, 且主要集中在前向传播过程当中。LoHan 利用这部分闲置带宽资源卸载更多计算过程中产生的激活值, 从而减少反向传播中所需重算的激活值。然而, 在 SSD 数量较少且硬件带宽无法被充分利用时, 反向传播的过程也受限于存储带宽, 卸载与加载更多的激活值只会进一步加剧系统 I/O 瓶颈。

3 系统设计

本节详细介绍本文提出的面向大模型训练的异构内存卸载与 I/O 优化方案 HiTrain 的核心设计与关键优化机制。

3.1 整体框架

针对当前大模型训练框架中普遍存在的显存不足、I/O 性能瓶颈以及计算资源利用率低等问题, 本文提出了 HiTrain。该方案围绕“高性能张量 I/O”与



(a) 反向传播 (b) 前向传播

Fig. 4 Storage-computing pipeline in training process

图 4 训练过程中的存储-计算流水线

“优化器流水线调度”两大核心维度展开设计，旨在实现对低成本大模型训练的高效支持。HiTrain的整体架构如图5所示，主要由张量存储和存储计算流水线调度2个核心模块构成。

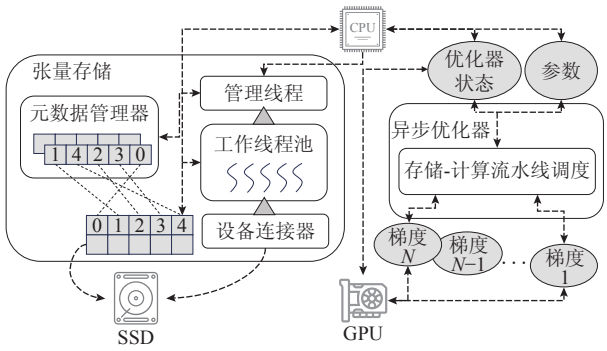


Fig. 5 Overall architecture diagram of HiTrain

图5 HiTrain整体架构图

1）基于SPDK的高性能张量存储模块。针对传统内核态I/O路径效率低下问题，引入SPDK用户态零拷贝I/O技术。该模块直接在用户态与NVMe SSD交互并读写张量，避免了内核路径的开销。其核心包括SPDK设备连接器、张量元数据管理器、SPDK管理线程以及高效并行的SPDK工作线程池，用于支撑大规模并发张量I/O请求的高吞吐处理，显著提高了NVMe SSD的利用效率与数据访问带宽。

2）基于异步优化器的存储-计算流水线调度模块。该模块重构了传统优化器的计算流程，通过引入梯度异步消费队列，实现了梯度计算与参数更新任务的并行化与异步化，有效减少流水线中的资源等待时间。同时，通过流水线重排的方式优先处理下一轮前向传播所需的前几层模型参数，进一步提升了计算资源的利用率和训练阶段的连续性。

通过上述2个模块的协同优化，HiTrain在保证训练精度的前提下，实现了GPU显存、CPU内存与NVMe SSD存储资源之间的高效协同调度，大幅缓解了大模型训练过程中的存储与计算瓶颈，从而高效支持面向单卡系统的百亿参数级模型训练任务。

3.2 基于SPDK的高性能张量存储模块

为了有效缓解大模型训练过程中频繁张量卸载带来的存储瓶颈问题，HiTrain在用户态管理SSD中的张量数据，从而避免了内核态I/O的开销。具体而言，我们设计并实现了一个基于SPDK的高性能I/O存储模块，作为系统“高性能张量I/O”优化路径的关键组成部分。

SPDK是一套用于加速对NVMe SSD设备访问的高性能存储开发工具包。其通过将NVMe设备驱动完全上移至用户空间，实现了应用程序与存储设备之间的直接数据通路，其旁路内核机制从根本上消除了用户态与内核态之间的切换开销以及传统I/O路径中冗长的层级处理流程[27]。与传统内核态I/O接口相比，SPDK具有更优的IOPS和更低的延迟[35]。此外，SPDK采用轮询机制主动检测I/O完成状态，避免中断处理延迟和其引入的性能抖动，更适合具有高频张量卸载的I/O负载场景。

为消除CPU内存与NVMe SSD卸载路径中的系统级瓶颈，HiTrain采用高性能的用户态SPDK接口替代Linux内核原生提供的异步I/O接口libaio，并重构了张量数据的交换机制，使得张量的加载与卸载可在用户态实现直接调度，在张量交换操作极为频繁的大模型训练场景中，显著降低I/O操作延迟，从而提升系统整体的训练吞吐量。

如图5所示，张量存储模块的整体架构由4个核心组件构成，分别是SPDK设备连接器、张量元数据管理器、SPDK管理线程和SPDK工作线程池。

SPDK设备连接器负责在系统初始化阶段构建SPDK NVMe运行环境。系统根据当前任务需求评估和分配若干大页（huge pages）内存资源，并通过PCIe地址完成NVMe控制器配对，之后即可通过SPDK驱动对目标设备进行高性能访问。

张量元数据管理器负责管理并维护卸载张量的元数据。其通过构建哈希表将张量ID映射至逻辑块起始地址，并在每次读写请求提交时完成地址检索。哈希表采用链地址法（separate chaining）以应对冲突，并支持动态扩容以适应张量规模增长带来的负载变化。

SPDK管理线程负责管理张量读写请求与SPDK工作线程池的任务分配。训练系统中异构数据调度层的多个NVMe交换器分别对应多个SPDK管理控制器对象，同时每个控制器对象又单独管理其所有SPDK工作线程池的生命周期，并与任务划分NVMe交换器通过SPDK管理线程提供的同步/异步读写接口提交I/O请求至SPDK工作线程池。另外，控制器还负责追踪读写任务执行流程和维护工作线程运行上下文。

SPDK工作线程池负责执行张量读写任务中具体的子任务，其设计要素主要包含线程同步、核心感知等。在执行具体的I/O任务时，SPDK管理线程将本次任务的通信缓冲区地址和逻辑块起始地址分发给工作线程池中所有线程的私有工作队列。此时工作线程池中的全部线程被唤醒，根据各自工作队列

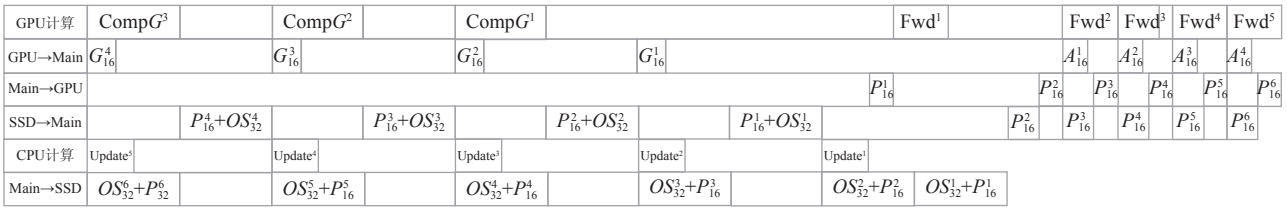
中的任务描述符进行拆分和执行，具体表现为所有工作线程并行执行原始任务的有序子任务。执行完毕后，任务描述符被添加至私有完成队列，并通知管理线程本次任务执行完毕。

张量存储模块为每个工作线程分配独立队列及对应的同步原语(即锁与条件变量)，并采用细粒度锁机制降低全局锁带来的线程间竞争开销。进一步，通过线程本地资源隔离，将全局共享队列的争用转化为线程局部资源的无锁访问，减少了同步操作的比较并交换(compare-and-swap)冲突概率。此外，模块采用了基于条件变量的事件驱动机制，通过精确的线程唤醒来降低由于忙等待带来的CPU空闲开销。

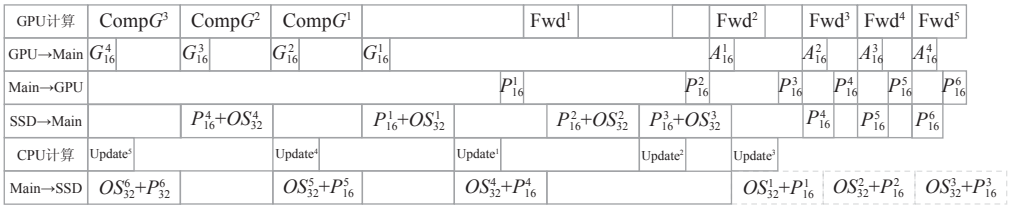
为充分发挥多核处理器的并行计算能力，我们采用了一种基于线程绑核的优化策略。具体而言，在当前训练系统中，将计算密集型任务(如CPU Adam Optimizer计算等)与I/O密集型任务进行物理隔离，分别绑定到同一处理器(或NUMA node)的不同核心上。其中，工作线程专门负责I/O任务，而管理线程则处理计算和控制任务。该机制避免了计算线程与I/O线程的资源竞争，有效减少了线程上下文切换开销，从而提升了系统整体吞吐量。

3.3 基于异步优化器的存储-计算流水线调度模块

在单卡训练环境中，尤其是采用NVMe SSD存储模型参数与优化器时，训练过程面临明显的存储-计算资源失衡问题。具体来说，如图6所示，在LoHan系统的训练流水线中，反向传播阶段主要受到SSD读写带宽的限制，呈现出显著的I/O瓶颈：GPU在完成每一层梯度计算后需等待梯度卸载以及参数与优化器状态加载等I/O操作，导致计算资源长期处于闲置状态。而在前向传播阶段，由于模型参数仅需以FP16精度加载，I/O负载较轻，计算操作成为主要性能瓶颈，导致SSD带宽浪费。



(a) LoHan流水线



(b) HiTrain流水线（本文）

Fig. 6 Comparison of pipeline optimization

图6 流水线优化对比

针对训练过程中出现的I/O与计算资源闲置问题，HiTrain提出了一种基于梯度异步消费机制的异步优化器存储-计算流水线优化策略。以打破GPU与CPU间的同步依赖，在确保计算正确性的同时，提升前向传播与反向传播间的I/O与计算的负载均衡性，最大化GPU与存储设备的并行利用率，从而提高系统整体训练效率。如图6(a)所示，在传统方案中，GPU在完成反向传播计算CompG^i后，需等待CPU与SSD间的I/O操作完成(即卸载优化器状态OS^6与加载OS^4)，才能继续后续的计算任务，形成多个周期的GPU空闲等待。通过梯度异步消费机制，GPU可将梯度快速写入CPU内存进行缓存，立即开始下一步计算CompG^{i+1}，无需等待CPU进行参数更新或SSD的I/O操作完成。如图6(b)HiTrain流水线所示，CPU优化器状态OS^4的I/O操作与GPU计算实现了解耦，缓解了图6(a)所示GPU因等待I/O操作串行所造成的计算资源浪费。

在GPU通过异步机制完成梯度卸载之后，HiTrain进一步对CPU端的优化器更新流水线进行了顺序重排与I/O操作延后处理，充分利用前向传播过程中闲置的I/O带宽，以进一步提升系统整体吞吐。如图6(b)所示，HiTrain不再按照梯度产生顺序处理所有参数更新，而是优先更新前向传播下一轮所需的前几层模型参数，确保GPU在下一轮前向传播开始时能

及时获取所需参数，减少 GPU 因参数更新延迟造成的空闲（即图 6 中 CompG^1 到 Fwd^1 之间的空闲）。同时，为进一步掩盖 I/O 操作延迟，HiTrain 将部分优化器状态的 I/O 操作（如图 6（b）中 OS^2 与 OS^3 等）延后至前向传播阶段中执行，充分利用前向传播中的 I/O 空闲带宽（如图 6（a）展示的前向传播 Fwd$^{3/4/5}$ 阶段中，读取模型参数 P_{16} 所需时间较短，SSD 带宽存在大量闲置周期）。这种将 I/O 操作跨阶段均摊的设计有效提升了 I/O 与计算流水线的重叠度，显著减少 GPU 空闲时间，提升了训练的连贯性与效率。

4 实验评估

为了验证 HiTrain 的有效性，本节通过对比实验对 HiTrain 中的高性能张量存储模块、优化器流水线调度模块以及整体训练性能进行系统性评估验证。

4.1 实验配置与测试指标

本实验在配备 NVIDIA RTX3090 和 RTX4090 两款消费级 GPU 的标准服务器平台上开展，具体实验环境配置如表 3 所示。为了评估 HiTrain 的高性能张量存储模块的 I/O 性能及其在大模型训练过程中的实际表现，我们选择平均带宽和训练的平均迭代时间作为性能测试指标。

Table 3 Experimental Environment Configuration

表 3 实验环境配置

配置	配置环境
CPU	Intel® Xeon® Silver 4 314 (2.40 GHz)
内存	2×256 GB DDR5 4800 MTps
SSD	Dell NVMe PM1743 RI E3.S+ Intel NVMe Datacenter SSD [Optane]
GPU	NVIDIA RTX3090/RTX4090
操作系统	Ubuntu 22.04.5 LTS (Kernel 5.15.0)
SPDK 版本	v24.09
CUDA	CUDA11.8
PyTorch	2.0.0

4.2 张量存储模块 I/O 性能评估

为了评估本文在 HiTrain 中设计并实现的基于 SPDK 的高性能张量存储模块的 I/O 性能，我们选择 psync（posix synchronous）[36] 和传统内存卸载方案中常见的基于 Linux 内核的 libaio 两种典型 I/O 接口作为对比对象，进行了不同 I/O 模块的读写性能测试。

首先为了对比不同 I/O 模块在不同文件块大小下的读写性能差异，本实验针对 psync, libaio, SPDK 这 3 种 I/O 接口，分别进行了 4 KB 和 1 MB 块大小的顺序读写性能测试，实验结果如图 7 所示。

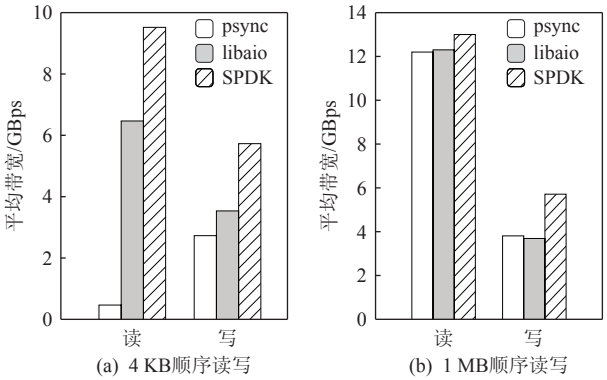


Fig. 7 Performance comparison of sequential read/write with different block sizes

图 7 不同块大小下的顺序读/写性能对比

从图 7（a）可见，在 4 KB 小块 I/O 场景下，SPDK 展现出明显性能优势，读写性能均显著优于 libaio 和 psync。SPDK 通过用户态 I/O 堆栈和轮询机制，避免了内核路径中的系统调用开销与中断处理延迟，尤其在小块 I/O 场景中能够充分释放 SSD 设备能力，体现出极佳的小文件处理性能。

在图 7（b）中，SPDK 依然保持领先性能，顺序读性能略优于 libaio 和 psync，顺序写性能优势更为明显，领先约 30%。这表明 SPDK 不仅在高频小块 I/O 中具备绝对优势，在大块张量卸载等典型训练负载场景下同样具备高吞吐特性，能够充分利用 NVMe SSD 的硬件带宽潜力。

整体而言，SPDK 在大小块 I/O 场景中均具备优越性能，特别是在小文件高频读写场景下优势更为显著，表明其在大模型训练过程中面对混合 I/O 模式（如 FP16 模型参数小张量+FP32 优化器状态大张量）时具备更好的适配性和扩展性，是大模型 I/O 优化的重要技术路径选择。

为了评估不同 I/O 模块对 CPU 核心数量的敏感性，本实验选取 psync, libaio, SPDK 这 3 种 I/O 接口，分别在单核和 8 核配置下进行了 1MB 顺序读写性能测试，实验结果如图 8 所示。

从图 8 可见，SPDK 在单核配置下已能接近 NVMe SSD 的物理带宽极限，顺序读写带宽表现均优于 libaio 和 psync，体现出用户态 I/O 栈低开销、高效直通设备的数据通路优势。由于单核下性能已接近设备上限，SPDK 在提升到 8 核后带宽增益相对有限，说明其 I/O 路径效率足够高，单核下即可充分利用硬件设备能力，具备良好的 CPU 资源利用率。

相比之下，psync 作为同步阻塞 I/O 接口，单核性

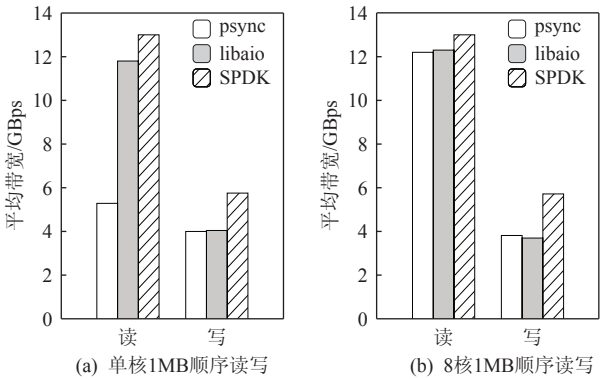


Fig. 8 Performance comparison of random read/write with different core numbers

图 8 不同核心数下的随机读/写性能对比

能受限明显, 随着核心数增加, 带宽提升幅度最大, 依靠 CPU 多并发能力弥补单线程 I/O 效率的不足。但这种方式需要较多 CPU 资源、调度开销显著、扩展性有限。libaio 在单核到 8 核间性能增幅较小, 受限于内核异步 I/O 接口的调度机制和中断处理效率, 整体未能充分发挥 SSD 高带宽潜力。

总体而言, SPDK 具备出色的单核性能与低 CPU 占用优势, 非常适合大模型训练过程中的高频张量卸载路径优化, 能够在保证高带宽利用率的同时, 释放更多 CPU 资源用于计算和调度; psync 和 libaio 在高性能卸载场景中存在明显瓶颈, 难以满足极限 I/O 性能需求。

为了进一步评估 HiTrain 中高性能张量存储模块在大模型训练中的 I/O 性能提升, 我们使用真实训练过程中的张量 I/O 请求对 LoHan 和 HiTrain 的张量存储模块进行了对比测试。如图 9 所示, 实验对比了 libaio, SPDK 在 1MB 顺序读写测试中的极限性能, 同时展示了 LoHan 和 HiTrain 在实际训练过程中的 I/O 性能表现。实验结果表明, 在实际训练负载中, 由于 LoHan 系统中采用的 libaio 接口在进行 I/O 操作时需要频繁切换内核态, 产生较高开销, 同时计算流水线调度和优化器更新均需要消耗 CPU 资源, 且两者相互竞争, 严重影响 I/O 效率, 导致 NVMe SSD 带宽无法被充分利用, 其平均读写带宽明显低于离线测试中的 libaio 峰值性能, 成为训练过程中的潜在瓶颈。而 HiTrain 系统中实现的基于 SPDK 的高性能张量存储模块完全在用户态执行, CPU 开销显著低于 libaio, 此外 HiTrain 通过将 I/O 模块绑定至指定核心进行 I/O 操作, 与 CPU 优化器更新所使用的核心隔离, 有效避免了计算与 I/O 流水线的相互干扰, 在实际模型训练中展现出优异的 I/O 性能, 其读写带宽接近离线 SPDK 的 1MB 顺序读写性能, 表明张量卸载路径能够高效调度 NVMe SSD 资源, 充分释放硬件带宽潜力。结合前述 1MB 随机读写实验, 进一步验证了 HiTrain 中所提出的存储优化设计具备良好的端到端性能表现, 可有效支撑大模型训练中高吞吐张量卸载需求, 显著优于现有方案。

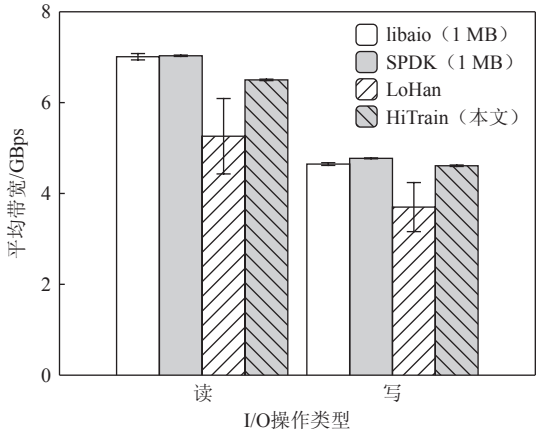


Fig. 9 Performance comparison of I/O module for different file blocks in actual training

图 9 实际训练中不同文件块 I/O 模块性能对比

4.3 HiTrain 整体训练性能评估

为进一步评估 HiTrain 方案在端到端训练场景下的整体性能表现, 本实验设计了 2 组对比测试: 第 1 组在固定批大小的情况下, 选取不同参数规模的 OPT-3 大模型作为训练负载以评估 HiTrain 在不同模型参数规模下的训练性能变化; 第 2 组在固定模型参数规模的情况下, 设置不同的批大小以进一步分析 HiTrain 在不同负载条件下的性能表现与可扩展性。实验对比对象包括: 1)含有基于 SPDK 高性能张量存储模块的 HiTrain 完整方案, 作为本文优化设计的主方案; 2)基于 libaio 实现张量卸载的 LoHan 方案, 作为现有 I/O 优化水平的对比对象; 3)CPU 卸载理想化方案, 通过模拟 CPU 内存容量无限充足的情况下将所有张量仅卸载于 CPU 内存中, 从而完全避免 SSD I/O 过程, 作为理论 I/O 最优参考线。通过以上实验方案对比, 能够有效衡量 HiTrain 系统在真实 I/O 环境下与理论最优性能之间的差距。

首先我们设置批大小为 64, 分别选取模型参数规模为 30 亿、70 亿、130 亿、300 亿的 OPT-3 大模型作为训练负载进行整体训练性能的对比测试, 实验结果如图 10 所示。从图 10 中可以看出, 随着模型参数规模的不断增大, HiTrain 系统整体性能显著优于 LoHan, 且性能优势随着模型规模增大而更加显著。在 300 亿参数规模下, HiTrain 单轮训练的时间相较

于 LoHan 缩短了 32.7%，整体训练吞吐量提升高达 1.49 倍。随着模型规模增大，LoHan 方案在频繁张量交换过程中的 I/O 瓶颈愈发突出，严重影响训练效率，而 HiTrain 通过高性能张量卸载路径与优化器流水线调度机制，有效提升了数据通路带宽利用率和存储计算重叠度，显著降低了 I/O 相关延迟。上述结果充分验证了 HiTrain 方案在低成本大模型训练场景下具备良好的可扩展性和吞吐性能优化能力，能够有效支撑在消费级 GPU 实现更大规模模型的高效训练需求。

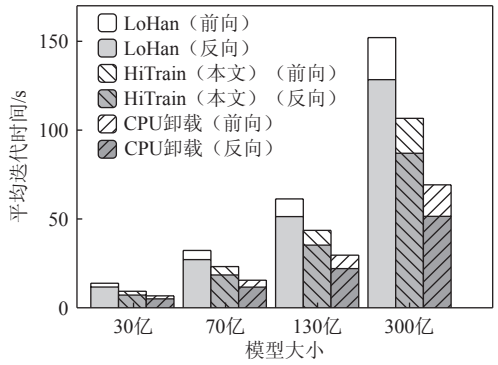


Fig. 10 Performance comparison under different model scales

图 10 不同模型规模下的训练性能对比

随后我们选取参数规模为 130 亿的 OPT-3 大模型作为训练负载，分别在批大小为 32, 64, 96, 128 的情况下进行了不同批大小的整体训练性能的对比测试，实验结果如图 11 所示，从图中可以看出，HiTrain 在不同批大小的情况下的性能均显著优于 LoHan，平均迭代时间平均减少超过 25%，展现出更稳定、更高效的训练吞吐性能。在批大小为 96 和 128 时，HiTrain 的单轮迭代时间已经接近基于 CPU 内存卸载的理想化方案，表明 HiTrain 方案能够充分释放 NVMe SSD 的带宽潜力。在训练过程中，随着批大小的增大，数据传输和计算负载同步加重，LoHan 方案 I/O 瓶颈进一步加剧，而 HiTrain 通过优化的数据调度与通信–计算重叠策略，保持了优异的可扩展性和训练效率，在批大小较大的场景下已具备接近基于内存卸载方案的训练吞吐。上述结果进一步验证了 HiTrain 在大模型大批量训练场景下具备优越的 I/O 性能和整体系统优化能力。

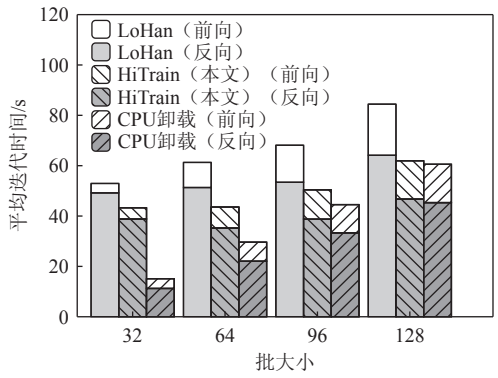


Fig. 11 Performance comparison of training with different batch sizes

图 11 不同批大小下的训练性能对比

综上，实验结果充分证明，本文提出的 HiTrain 方案在低成本大模型训练场景中，对于不同模型规模和不同批大小的训练任务均能显著提升内存卸载 I/O 带宽和系统整体训练效率，训练吞吐性能接近理论最优参考线，具备良好的通用性和可扩展性，能够有效支撑未来在普通硬件设备上进行超大规模模型的高效训练需求。

5 总结展望

本文提出了一种面向大模型训练的异构内存卸载与 I/O 优化方案 HiTrain，设计并实现了基于 SPDK 的高性能张量存储模块，有效避免了内核 I/O 路径带来的系统调用与中断开销。同时基于梯度异步消费与优化器流水线调度，实现了 GPU 反向传播与 CPU 参数更新阶段的高效重叠。实验证明，在不同参数规模的模型训练中，HiTrain 在训练吞吐、GPU 利用率与 I/O 延迟方面均显著优于现有方案，展现出良好的性能与可扩展性。未来将进一步探索多 NVMe SSD 与 NUMA 架构下的高效张量调度策略，结合运行时系统进行 I/O 计算协同调度，并拓展本系统在中大型集群与边缘训练环境中的部署能力，以推动大模型训练系统在不同硬件资源约束下的广泛适用性与性能优化。

作者贡献声明：汤志航、兰颢提出论文思路，负责方案的设计实施以及论文撰写；刘政国参与方案实现和实验验证；汪庆对方案设计和论文撰写提出修改意见；赵来平、李克秋对论文选题和论文撰写提出指导意见。

参 考 文 献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J/OL]. Advances in Neural Information Processing Systems, 2017[2025-06-02]. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[2] Achiam J, Adler S, Agarwal S, et al. GPT−4 technical report[J]. arXiv

preprint, arXiv: 2303.08774, 2023

[3] Jiang Juyong, Wang Fan, Shen Jiasi, et al. A survey on large language models for code generation[J]. arXiv preprint, arXiv: 2406.00515, 2024

[4] Wu Jiayu, Gan Wensheng, Chen Zefeng, et al. Multimodal large language models: A survey[C]//Proc of the 2023 IEEE Int Conf on Big Data (BigData). Piscataway, NJ: IEEE, 2023: 2247−2256

[5] Feng Yangyang, Wang Qing, Xie Minhui, et al. From BERT to ChatGPT: Challenges and technical development of storage systems for large model training[J]. Journal of Computer Research and Development, 2024, 61(4): 809−823 (in Chinese)
(冯杨洋, 汪庆, 谢旻晖, 等. 从 BERT 到 ChatGPT: 大模型训练中的存储系统挑战与技术发展[J]. 计算机研究与发展, 2024, 61(4): 809−823)

[6] NVIDIA. NVIDIA H200 tensor core GPU[EB/OL]. Santa Clara, CA: NVIDIA Corporation, 2023 (2024-05-26)[2025-06-03]. https://www.nvidia.com/en-us/data-center/h200/

[7] Touvron H, Martin L, Stone K, et al. LlaMA 2: Open foundation and fine-tuned chat models[J]. arXiv preprint, arXiv: 2307.09288, 2023

[8] Luo Xudong, Deng Zhiqi, Yang Bingxia, et al. Pre-trained language models in medicine: A survey[J]. Artificial Intelligence in Medicine, 2024, 154: 102904

[9] Li Junyi, Tang Tianyi, Wayne X Z, et al. Pre-trained language models for text generation: A survey[J]. ACM Computing Surveys, 2024, 56(9): 1−39

[10] Rasley J, Rajbhandari S, Ruwase O, et al. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters[C]//Proc of the 26th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining. New York: ACM, 2020: 3505−3506

[11] Shoeybi M, Patwary M, Puri R, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism[J]. arXiv preprint, arXiv: 1909.08053, 2019

[12] Park Y M, Ahn S Y, Lim E J, et al. Deep learning model parallelism[J]. Electronics and Telecommunications Trends, 2018, 33(4): 1−13

[13] Fan Shiqing, Rong Yi, Meng Chen, et al. DAPPLE: A pipelined data parallel approach for training large models[C]//Proc of the 26th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2021: 431−445

[14] Liu Aixin, Feng Bei, Xue Bing, et al. DeepSeek-v3 technical report[J]. arXiv preprint, arXiv: 2412.19437, 2024

[15] Li Shenggui, Liu Hongxin, Bian Zhengda, et al. Colossal-AI: A unified deep learning system for large-scale parallel training[C]//Proc of the 52nd Int Conf on Parallel Processing. New York: ACM, 2023: 766−775

[16] Pfister G. An introduction to the InfiniBand architecture[J]. High Performance Mass Storage and Parallel I/O, 2001, 42(102): 617−632

[17] Foley D, Danskin J. Ultra-performance Pascal GPU and NVLink interconnect[J]. IEEE Micro, 2017, 37(2): 7−17

[18] Bae J, Lee J, Jin Y, et al. FlashNeuron: SSD-enabled large-batch training of very deep neural networks[C]//Proc of the 19th USENIX Conf on File and Storage Technologies (FAST'21). Berkeley: CA, USENIX Association, 2021: 387−401

[19] Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: Memory optimizations toward training trillion parameter models[C]//Proc of the 2020 Int Conf for High Performance Computing, Networking, Storage and Analysis(SC'20). Piscataway, NJ: IEEE, 2020: 1−16

[20] Ren Jie, Rajbhandari S, Aminabadi R Y, et al. ZeRO-Offload: Democratizing billion-scale model training[C]//Proc of the 2021 USENIX Annual Technical Conf (USENIX ATC'21). Berkeley: CA USENIX Association, 2021: 551−564

[21] Rajbhandari S, Ruwase O, Rasley J, et al. ZeRO-Infinity: Breaking the gpu memory wall for extreme scale deep learning[C]//Proc of the 2021 Int Conf for High Performance Computing, Networking, Storage and Analysis (SC'21). Piscataway, NJ: IEEE, 2021: 1−14

[22] Liao Changyue, Sun Mo, Yang Zihan, et al. LoHan: Low-cost high-performance framework to fine-tune 100 B model on a consumer GPU[J]. arXiv Preprint, arXiv: 2403.06504, 2024

[23] Yang Ziye, Harris J R, Walker B, et al. SPDK: A development kit to build high performance storage applications[C]//Proc of the 2017 IEEE Int Conf on Cloud Computing Technology and Science (CloudCom2017). Piscataway, NJ: IEEE, 2017: 154−161

[24] Thompson A, Newburn C J. GPUDirect storage: A direct path between storage and GPU memory[EB/OL]. NVIDIA Developer Whitepapers, 2019[2025-06-05]. https://developer.nvidia.com/blog/gpudirect-storage

[25] Wu Kun, Park J B, Zhang X, et al. SSDTrain: An activation offloading framework to SSDs for faster large language model training[J]. arXiv preprint, arXiv: 2408.10013, 2024

[26] Zhang Haoyang, Zhou Yirui, Xue Yuqi, et al. G10: Enabling an efficient unified GPU memory and storage architecture with smart tensor migrations[C]//Proc of the 56th Annual IEEE/ACM Int Symp on Microarchitecture. Piscataway, NJ: IEEE, 2023: 395−410

[27] Xu Dong, Feng Yuan, Shin K, et al. Efficient tensor offloading for large deep-learning model training based on compute express link[C]//Proc of the 2024 Int Conf for High Performance Computing, Networking, Storage and Analysis (SC'24). Piscataway, NJ: IEEE, 2024: 1−18

[28] Didona D, Pfefferle J, Ioannou N, et al. Understanding modern storage APIs: A systematic study of libaio, SPDK, and io_uring[C]//Proc of the 15th ACM Int Systems and Storage Conf. New York: ACM, 2022: 120−127

[29] Ravi J, Byna S, Koziol Q, et al. Evaluating asynchronous parallel I/O on HPC systems[C]//Proc of the 37th IEEE Int Parallel and Distributed Processing Symp (IPDPS'23). Piscataway, NJ: IEEE, 2023: 211−221

[30] Hao Dongdong, Gao Congming, Shu Jiwu. Design of user space storage architecture for SCSI subsystem[J]. Journal of Computer Research and Development, 2025, 62(3): 633−647 (in Chinese)
(郝栋栋, 高聪明, 舒继武. 面向 SCSI 子系统的用户空间存储架构设计[J]. 计算机研究与发展, 2025, 62(3): 633−647)

[31] Li Junzhen, Pan Xiurui, Yi Shushu, et al. A survey on user-space storage and its implementations[J]. arXiv preprint, arXiv: 2306.10503, 2023

[32] Kim H, Lee Y, Kim J S. NVMeDirect: A user-space I/O framework for application-specific optimization on NVMe SSDs[C]//Proc of the 8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'16). Berkeley, CA: USENIX Association, 2016: 1−5

[33] The Linux Foundation. I/O latency optimization with polling [R]. San Francisco, CA: The Linux Foundation, 2017

[34] Huang Jian, Badam A, Qureshi M K, et al. Unified address translation for memory-mapped SSDs with FlashMap [C] //Proc of the 42nd Annual Int Symp on Computer Architecture. New York: ACM, 2015: 580−591

[35] Gerhorst L. Analysis of interrupt handling overhead in the Linux kernel [D]. Erlangen: Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 2018

[36] Ren Zebin, Trivedi A. Performance characterization of modern storage stacks: posix I/O, libaio, spdk, and io_uring[C]//Proc of the 3rd Workshop on Challenges and Opportunities of Efficient and Performant Storage Systems. New York: ACM, 2023: 35−45



Tang Zhihang, born in 1993. PhD candidate, assistant researcher. Member of CCF. His main research interests include distributed computing and large model system.

汤志航, 1993 年生。博士研究生，助理研究员。CCF 会员。主要研究方向为分布式计算、大模型系统。



Lan Hao, born in 1993. PhD, assistant researcher. Member of CCF. His main research interest includes AI systems and algorithms.

兰　颢, 1993 年生。博士，助理研究员。CCF 会员。主要研究方向为人工智能系统与算法。



Liu Zhengguo, born in 1999. Master candidate. Student member of CCF. His main research interests include AI systems and storage systems.

刘政国, 1999 年生。硕士研究生。CCF 学生会员。主要研究方向为人工智能系统、存储系统。



Wang Qing, born in 1997. PhD, assistant researcher. Member of CCF. His main research interests include storage systems and memory systems.

汪　庆, 1997 年生。博士，助理研究员。CCF 会员。主要研究方向为存储系统、内存系统。



Zhao Laiping, born in 1985. PhD, professor, PhD supervisor. Member of CCF. His main research interest includes cloud computing system.

赵来平, 1985 年生。博士，教授，博士生导师。CCF 会员。主要研究方向为云计算系统。



Li Keqiu, born in 1971. PhD, professor, PhD supervisor. Fellow of CCF. His main research interests include computer networks and cloud computing.

李克秋, 1971 年生。博士，教授，博士生导师。CCF 会士。主要研究方向为计算机网络、云计算。