PR #1793 完整报告

radixark/miles

feat(optimizer): NVMe optimizer-state streaming as a miles plugin

合并时间：2026-08-01 13:57

原文链接：http://prhub.com.cn/radixark/miles/pull/1793

执行摘要

- 一句话：NVMe 优化器状态流式插件，step 峰值显存降约 52%
- 推荐动作：值得精读。该 PR 展示了跨仓库功能下沉的完整工程范式：契约方法收敛、启动期守卫替代运行期故障、以及“以 assert 把静默错误变成响亮错误”的防御思路。尤其推荐关注 _Stager.transfer 的跨 dtype 设备中转设计、plan_buckets 与 DDP 解耦的动机，以及 actor_factory 对 torch_memory_saver 二进制的字节级检查——这类细节是大型 RL 训练系统稳定性的关键。

功能与动机

PR body 明确指出：--offload-train-target=disk（#1575）只能在阶段边界 spill 暂停的 actor，"That cannot help when the optimizer state does not fit the GPU while the step runs — by the time the Adam kernel launches, everything is resident again"。当优化器状态在 step 运行期间就放不下 GPU 时，offload 无能为力；streaming 才是解法。该机制先在 Megatron-LM#63 完成验证（GLM-5.2 744B 8xGB300、Qwen3.5-35B-A3B 8xH200），本 PR 把它移入 miles 并让 Megatron 侧只剩两个 checkpoint 钩子。

实现拆解

1. 实现迁移与实例绑定：将 Megatron-LM#63 的 megatron/core/optimizer/nvme_state_store.py 迁入 miles_plugins/optimizers/nvme_stream.py（+522 行），并把 DistributedOptimizer 内五处 if self._nvme_state_store is not None 分支改为 _bind() 时的实例绑定：step_with_ready_grads 绑到 store.step()（含参数 all-gather 尾部）、reload_model_params 路由到 store.refresh_main_from_model_params、state_dict/load_state_dict/sharded_state_dict 返回空，由 save_to/load_from 承载真实字节。miles 侧在 miles/backends/megatron_utils/model.py 的 setup_model_and_optimizer 中通过两个 lazy import 完成装配，Megatron 侧最终只保留 checkpointing.py 里的一对钩子。
2. 参数面与静默失效防御：在 miles/utils/arguments.py 新增 --stream-optimizer-state-to-disk 与 --stream-optimizer-state-moment-dtype（默认 fp32，可选 bf16/fp16/fp8e4m3/fp8e5m2，fp8 不推荐），并复用 --offload-train-disk-dir/--offload-train-disk-chunk-mb（两机制各自取子目录）。校验块点名拒绝所有会读到空 storage/ 空 state 的开关：reset_optimizer_states、save_local_weight_checksum、enable_witness，以及互斥项 optimizer_cpu_offload、

offload_optimizer_states、precision-aware optimizer、单 LoRA 与 multi-LoRA、--indep-dp（rank 目录跨 cell 冲突），并强制要求 --offload-train-target=disk。

3. 流式存储核心：NVMeOptimizerStateStore 按 DDP bucket 聚合 entry，再用 plan_buckets 按 200M numel 上限切分，与 DDP 大 bucket 解耦；每个 _Bucket 拥有独立文件（main/exp_avg/exp_avg_sq 三段按 4096 对齐）、独立克隆的 FusedAdam 与 I/O 偏移表。step() 逐个 bucket 执行"物化 storage -> 读入 -> 逐 bucket Adam 更新 -> 更新后的主参数拷回参数缓冲 -> 写回 -> storage 释放"。首次 fetch 只读 main，moment 由 FusedAdam 惰性创建后再落盘；_Stager 用固定大小 pinned 缓冲中转，跨 dtype 拷贝先在设备侧转换（bf16 约快 40 倍、fp8 约快 100 倍）。目录启动时清除、退出时 atexit 回收。
4. 正确性护栏与配套：miles/ray/train/actor_factory.py 对 LD_PRELOAD 的 torch_memory_saver 二进制做字节级断言（必须含 TMS_INIT_ENABLE_DISK_BACKUP），防止旧版二进制忽略 disk 变量、pause 后无备份、resume 返回未初始化内存，最终在 rollout 中表现成 NaN/grad-norm 故障。新增 tests/e2e/megatron/test_qwen3_4B_offload_disk_stream.py，除完整 colocate RL 循环外，还逐 rank 断言 worker 日志出现 NVMe streaming step:，阻止"开关丢失但训练照常完成"的假阳性；tests/ci/labels.py 新增 miles-plugin 标签，LoRA 测试夹具补 stream_optimizer_state_to_disk=False，docs/advanced/disk-offload.md 新增 streaming 章节（含同拓扑 resume、同步 checkpoint、fp8 不推荐等限制）。

关键文件：

- miles_plugins/optimizers/nvme_stream.py（模块 优化器插件；类别 source；类型 dependency-wiring；符号 _Entry, _align, _resize, _allocate_file）：本 PR 的核心：NVMe 优化器状态流式存储的完整实现，包含 _Stager 中转、_Bucket 生命周期、checkpoint 四方法契约与所有构造守卫。
- tests/e2e/megatron/test_qwen3_4B_offload_disk_stream.py（模块 E2E 测试；类别 test；类型 test-coverage；符号 prepare, _assert_offloaded_to_disk, _assert_streamed, execute）：新增 E2E 测试：验证流式与磁盘 offload 组合在真实 colocate RL 循环中工作，并逐 rank 断言流式真的发生，防静默失效。
- miles/utils/arguments.py（模块 参数校验；类别 source；类型 dependency-wiring）：参数面接入点：新增两个 switch、扩展两个既有磁盘参数语义，并集中书写整套互斥与前置条件断言，是静默失效防御的主战场。
- miles/backends/megatron_utils/model.py（模块 模型装配；类别 source；类型 data-contract）：装配入口：在 setup_model_and_optimizer 中按开关 lazy import 并调用 setup_optimizer_state_streaming，是 miles 侧与 Megatron 的唯一接缝。
- miles/ray/train/actor_factory.py（模块 Actor 工厂；类别 source；类型 dependency-wiring）：环境守卫：对 LD_PRELOAD 的 torch_memory_saver 二进制做字节检查，防止旧版无 disk 后端导致 resume 后内存未初始化、最终表现为模型侧 NaN 故障。
- miles_plugins/optimizers/__init__.py（模块 插件入口；类别 source；类型 core-logic）：新插件包的入口文件，随核心模块一起新增，标识 miles_plugins/optimizers 命名空间的建立。

- tests/fast/backends/megatron_utils/test_lora_model_branches.py（模块 LoRA 测试；类别 test；类型 test-coverage）：测试夹具适配：为构造 args 的 helper 补充 stream_optimizer_state_to_disk=False，保证 LoRA 分支测试不受新校验影响。
- docs/advanced/disk-offload.md（模块 高级文档；类别 docs；类型 documentation）：功能文档：新增 optimizer-state streaming 章节，给出用法、位等价语义、bf16 建议、同拓扑 resume/ 同步 checkpoint/fp8 不推荐三项限制，以及两机制叠加的收益说明。
- tests/ci/labels.py（模块 CI 标签；类别 test；类型 test-coverage）：CI 配套：新增 miles-plugin 标签，用于路由本 PR 引入的插件 E2E 测试。

关键符号：setup_optimizer_state_streaming, NVMeOptimizerStateStore.init, NVMeOptimizerStateStore.step, NVMeOptimizerStateStore.refresh_main_from_model_params, NVMeOptimizerStateStore.save_to, NVMeOptimizerStateStore.load_from, _Bucket.fetch, _Bucket.flush, _Bucket.materialize_main, _Bucket.allocate_moments, _Stager.transfer, plan_buckets, _rw_full, _allocate_file, _build_fp32_optimizer, _copy_main_to_model_params

关键源码片段

miles_plugins/optimizers/nvme_stream.py

本 PR 的核心：NVMe 优化器状态流式存储的完整实现，包含 _Stager 中转、_Bucket 生命周期、checkpoint 四方法契约与所有构造守卫。

```
# NVMe 优化器状态流式：fp32 主参数与 Adam 动量按 bucket 存于节点本地文件。
# step() 逐个 bucket 处理：物化 storage -> 读入 -> 逐 bucket FusedAdam 更新 ->
# 更新后的主参数拷回参数缓冲 -> 写回 -> storage 释放，峰值驻留被限在一个 bucket 内。

class _Stager:
    # 固定大小 pinned 主机缓冲作为 GPU <-> NVMe 中转站，
    # 无论一次搬移多少数据，主机内存占用都有上界。
    def __init__(self, nbytes: int):
        self._buf = torch.empty(nbytes, dtype=torch.uint8, pin_memory=True)
        self._bytes = self._buf.numpy()
        self._device_buf = None

    def _device_staging(self, dtype, numel, like):
        # 跨 dtype 的 GPU <-> pinned 主机拷贝不会走 DMA 路径；
        # 先在设备侧完成转换、再搬运同 dtype 字节，bf16 约快 40 倍、fp8 约快 100 倍。
        size = self._buf.numel()
        if self._device_buf is None or self._device_buf.device != like.device:
            self._device_buf = torch.empty(size, dtype=torch.uint8, device=like.device)
        return self._device_buf[: numel * dtype.itemsize].view(dtype)

    def transfer(self, fd, offset, tensor, dtype, *, to_disk):
        flat = tensor.view(-1)
        cast = dtype != flat.dtype
        chunk = self._buf.numel() // dtype.itemsize
        pos = 0
```

```
        while pos < flat.numel():
            numel = min(chunk, flat.numel() - pos)
            host = self._buf[: numel * dtype.itemsize].view(dtype)
            at = offset + pos * dtype.itemsize
            nbytes = numel * dtype.itemsize
            if to_disk:
                # 写盘：必要时先在设备侧转换 dtype，再经 pinned 缓冲 pwritev 落盘
                if cast:
                    staged = self._device_staging(dtype, numel, flat)
                    staged.copy_(flat[pos:pos + numel])
                    host.copy_(staged)
                else:
                    host.copy_(flat[pos:pos + numel])
                _rw_full(os.pwritev, fd, at, self._bytes[:nbytes])
            else:
                # 读回：preadv 到 pinned 缓冲，再拷回 GPU（同样先转换 dtype）
                _rw_full(os.preadv, fd, at, self._bytes[:nbytes])
                if cast:
                    staged = self._device_staging(dtype, numel, flat)
                    staged.copy_(host)
                    flat[pos:pos + numel].copy_(staged)
                else:
                    flat[pos:pos + numel].copy_(host)
            pos += numel
        return flat.numel() * dtype.itemsize


class _Bucket:
    # 一个 bucket 是流式的最小工作单元：一个文件、一段依次排布的段布局
    #  （main / exp_avg / exp_avg_sq），以及为它独立克隆的 FusedAdam 实例。

    def fetch(self) -> int:
        # 首次 fetch 只读 main：moment 由 FusedAdam 在 step 1 惰性创建，
        # 未就绪前文件中还没有 exp_avg / exp_avg_sq 段的数据。
        return self._move(SEGMENTS if self.moments_ready else SEGMENTS[:1], to_disk=False)

    def flush(self, segments=SEGMENTS) -> int:
        # 写回后立即把 storage 释放为 0 大小，GPU 驻留回到只占一个 bucket 的量级。
        moved = self._move(segments, to_disk=True)
        self.moments_ready = self.moments_ready or tuple(segments) == SEGMENTS
        return moved

    def _move(self, segments, *, to_disk):
        # 上载（读回）前先为 storage 分配空间；卸载（写盘）后调用 _resize 释放。
        # 跨 dtype 场景（如 bf16 moment）由 _Stager.transfer 内部的设备中转完成转换。
        moved = 0
        for segment in segments:
            for tensor, offset in self._tensors(segment):
                if not to_disk:
```

```
                _resize(tensor, tensor.numel())
            moved += self._stager.transfer(
                self.fd, offset, tensor, self.dtypes[segment], to_disk=to_disk
            )
            if to_disk:
                _resize(tensor, 0)
    return moved
```

tests/e2e/megatron/test_qwen3_4B_offload_disk_stream.py

新增 E2E 测试：验证流式与磁盘 offload 组合在真实 colocate RL 循环中工作，并逐 rank 断言流式真的发生，防静默失效。

```
# ── E2E 断言：streaming 必须真的发生，而不是静默未启用 ──
# 跑完训练只是半个检查：如果开关在传向 actor 途中丢失、或插件导入失败被吞掉，
# 训练照样能完成。因此这里逐 rank 核对 worker 日志中是否出现真实的
# "NVMe streaming step:" 记录，缺一个 rank 就判失败。
def _assert_streamed():
    logs = glob.glob("/tmp/ray/session_latest/logs/worker-*")
    assert logs, "no Ray worker logs to check for the streaming path"

    streamed = set()
    for path in logs:
        with open(path, errors="ignore") as f:
            if any("NVMe streaming step:" in line for line in f):
                streamed.add(path)

    assert len(streamed) == NUM_GPUS, (
        f"expected {NUM_GPUS} ranks to log streaming steps, saw {len(streamed)}"
    )
    print(f"optimizer state streaming ran on {len(streamed)} ranks")
```

miles/utils/arguments.py

参数面接入点：新增两个 switch、扩展两个既有磁盘参数语义，并集中书写整套互斥与前置条件断言，是静默失效防御的主战场。

```
# ── 参数校验：streaming 的静默失效防御 ──
# 流式开启后，主参数 storage 会在 step 之间被释放为 0，master optimizer 的
# 状态也由 store 持有；所有直接读这些位置的开关都必须在这里被点名拒绝，
# 否则它们会读到空值而静默返回，而不是报错。
if args.stream_optimizer_state_to_disk:
    # 只与 --offload-train-target=disk 成对部署：两者回答同一个压力
    assert args.offload_train_target == "disk", (
        "--stream-optimizer-state-to-disk requires --offload-train-target=disk"
    )
    assert args.use_distributed_optimizer, "requires the distributed optimizer"
    assert args.optimizer == "adam", "--optimizer adam required"
    assert not (args.multi_lora or is_lora_enabled(args)), (
        "LoRA checkpoint 路径绕过 Megatron 的 save/load，store 的 save_to/load_from 不会触发"
    )
```

```
assert not args.optimizer_cpu_offload, "mutually exclusive with CPU offload"
assert not args.offload_optimizer_states, "mutually exclusive with --offload-optimizer-states"
assert not args.use_precision_aware_optimizer, "fp32 main params must be held by mcore"
assert not args.reset_optimizer_states, "reset would silently do nothing"
assert not args.save_local_weight_checksum, "param.main_param storage is resized to 0"
assert not args.enable_witness, "witness reads the master optimizer per-param state"
```

评论区精华

核心讨论集中在三点：

- 合并顺序：Zhichenzzz 在 PR 上留言 “I will retarget this PR to main and merge #1575 first”，先落基础 PR 再合本 PR，最终按此执行。
- 功能扩展建议：Zhichenzzz 在 approve 时提问 “qq: could we also add the cpu-offload streaming as the choice?”，即是否把流式优化器状态也做成 CPU 备份选项；该建议未在本 PR 中实现，当前设计强制与磁盘 offload 成对部署。
- 非侵入性确认：fzyzcjy 受邀 review，表示 “not check in detail, but looks not invasive and thus lgtm”。
- 合并顺序与 base 分支调整 (other): 按该计划执行，最终以 main 为 base 完成合并。
- 是否补充 CPU offload 流式选项 (design): PR 中未新增该选项；当前设计强制 --stream-optimizer-state-to-disk 依赖 --offload-train-target=disk，CPU 流式未被实现，属于可能的后续扩展。
- 非侵入性确认 (other): 通过，未提出具体修改意见。

风险与影响

- 风险：
  - 跨仓库契约风险：_copy_main_to_model_params 以 commit+ 行区间注释锁定 Megatron 侧实现，Megatron 升级后可能漂移；代码里 assert world_range.size == shard_main_param.nelement() 作为 tripwire，但若上游重构导致该断言被绕过，fp8 权重更新可能被跳过且无报错。
  - 回归风险：store 会 _resize(main_param, 0) 释放 storage，任何未纳入守卫的路径（如未来新增的 main_param 读取者）都会静默读到空值；本 PR 只能覆盖当下已知的 reset_optimizer_states、save_local_weight_checksum、enable_witness。
  - 性能风险：步进时间明显变长，8xH200 实测 actor_train +39%（bf16 moments），checkpoint save 是同步的、绕过 --async-save，保存大模型时阻塞时长不可忽略。
  - 兼容性风险：同拓扑 resume only，改变 TP/PP/DP/EP 会直接 layout assert 失败；开启流式后无法恢复启用流式之前写的 checkpoint（需 --no-load-optim 接受全新优化器状态）。
  - 环境风险：--indep-dp 下 rank 目录没有 cell 命名空间，两个 cell 共享路径会互相清除 bucket 文件，已在启动期拒绝；torch_memory_saver 旧二进制无 disk 后端时会在 actor_factory 启动即失败，而不是 mid-run 才暴露。
- 影响：

- 用户影响：面向 DP=1 大模型 RL 训练（如 GLM-5.2 744B 8xGB300），流式开启后峰值显存大幅下降（Qwen3-30B-A3B 实测 -42.3 GB/-52%），默认 fp32 位等价保证结果不变，bf16 是显式的性能 opt-in。
- 系统影响：新增 miles_plugins/optimizers 插件目录与 miles-plugin CI 标签；与 Megatron-LM#63 形成四方法 duck-typed 契约，Megatron 侧代码缩减到 +19 行；miles/utils/arguments.py 的互斥校验成为新功能的准入清单。
- 团队影响：确立了 miles 与上游 Megatron 的协作模式（pinned commit 注释 + 断言 tripwire + 显式契约方法），后续任何触碰 optimizer state 的新特性都需对照此清单补齐守卫。
- 风险标记：核心优化器路径变更，跨仓库契约依赖，同拓扑 resume 限制，step 耗时增加，checkpoint save 同步阻塞，fp8 存储不受推荐

关联脉络

- PR #1575 offload: paused-actor disk offload (--offload-train-target=disk): PR body 与 reviewer 评论指明本 PR 构建在 #1575 之上（#1575 标题未在提供的上下文中直接给出，此处按 PR body 描述概括）：streaming 复用其 --offload-train-disk-dir/--offload-train-disk-chunk-mb 参数与磁盘布局，且必须与其成对部署。
- PR #63 feat(optimizer): NVMe streaming of DistributedOptimizer state: Megatron-LM 侧的配套实现（跨仓库），本 PR 把该实现从 Megatron 移入 miles_plugins/optimizers/nvme_stream.py，并与它形成四方法 duck-typed 契约；PR body 明确 'Pairs with radixark/Megatron-LM#63'。