PR #6193 完整报告

verl-project/verl

[megatron] fix: avoid 2x peak host memory on Megatron model offload

合并时间：2026-04-30 14:24

原文链接：http://prhub.com.cn/verl-project/verl/pull/6193

执行摘要

- 一句话：修复 Megatron 模型 CPU 卸载时峰值内存翻倍问题
- 推荐动作：值得精读。该 PR 清晰展示了 PyTorch pinned memory 分配的内存管理陷阱，并提供了简洁有效的修复模式（惰性初始化 + 断言保护），适用于类似场景。建议合并。

功能与动机

训练 Qwen3-235B-A22B 时在 step 2/3 发生 OOM（cgroup 限制 1700 GiB），调试发现 offload_megatron_model_to_cpu 中每次卸载都通过 buffer.param_data.cpu() .pin_memory() 重新分配 pinned host buffer，RHS 求值期间新旧 block 共存导致瞬时峰值 2x ≥2600 GiB，超出限制。作者在 PR body 中明确描述了该机制。

实现拆解

1. 核心变更：在 verl/utils/megatron_utils.py 的 offload_megatron_model_to_cpu 函数中，将原来每轮都执行 buffer.param_data.cpu() .pin_memory() 替换为 惰性一次性分配——通过 getattr 检查 cpu_data 是否已存在，若不存在则 torch.empty(size, dtype, pin_memory=True) 创建，然后通过 copy_(non_blocking=False) 同步拷贝数据。
2. 不变性断言：在 cpu_data 已存在时，断言 shape 和 dtype 与当前 param_data 一致，防止静默重新引入 2x 峰值。
3. 同步拷贝：拷贝必须同步完成，确保 resize_(0) 释放 GPU 存储前数据已全部复制到 CPU。
4. 无 API 变化：对外接口完全一致，仅内部内存管理策略优化。

关键文件：

- verl/utils/megatron_utils.py（模块 工具库；类别 source；类型 core-logic；符号 offload_megatron_model_to_cpu）：核心修复文件：修改了 offload_megatron_model_to_cpu 函数的内存分配逻辑，每 DDP buffer 最多分配一次 pinned buffer。

关键符号：offload_megatron_model_to_cpu

关键源码片段

verl/utils/megatron_utils.py

核心修复文件：修改了 offload_megatron_model_to_cpu 函数的内存分配逻辑，每 DDP buffer 最多分配一次 pinned buffer。

```
@torch.no_grad()
def offload_megatron_model_to_cpu(models):
    for model_chunk in models:
        if isinstance(model_chunk, DDP):
            model_chunk_all_buffers = [model_chunk.buffers, model_chunk.expert_parallel_buffers]
            for buffers in model_chunk_all_buffers:
                for buffer in buffers:
                    if buffer.param_data.storage().size() > 0:
                        # 惰性分配：每个 DDP buffer 仅分配一次 pinned cpu_data
                        # 原实现每轮重新分配，导致新旧 pinned block 共存，峰值内存翻倍
                        existing = getattr(buffer.param_data, "cpu_data", None)
                        if existing is None:
                            buffer.param_data.cpu_data = torch.empty(
                                buffer.param_data.size(),
                                dtype=buffer.param_data.dtype,
                                device="cpu",
                                pin_memory=True,
                            )
                            buffer.param_data_size = buffer.param_data.storage().size()
                        else:
                            # 断言 shape 和 dtype 一致，防止静默重新分配引入 2x 峰值
                            assert existing.shape == buffer.param_data.shape, (
                                f"cpu_data shape {tuple(existing.shape)} != "
                                f"param_data shape {tuple(buffer.param_data.shape)}; "
                                "reallocating would reintroduce the 2x peak."
                            )
                            assert existing.dtype == buffer.param_data.dtype, (
                                f"cpu_data dtype {existing.dtype} != "
                                f"param_data dtype {buffer.param_data.dtype}; "
                                "reallocating would reintroduce the 2x peak."
                            )
                        # 同步 D2H 拷贝：确保拷贝完成后再释放 GPU 存储
                        buffer.param_data.cpu_data.copy_(
                            buffer.param_data.data, non_blocking=False
                        )
                        buffer.param_data.storage().resize_(0)
                    assert buffer.param_data_size == buffer.param_data.cpu_data.storage().size()
                    # ... 后续 grad_data offload 不变
```

评论区精华

- Begunner指出原实现并非“持续内存泄漏”（freed pinned memory 可复用），而是 allocate-before-free 导致的瞬时 2x 峰值。作者接受该指正，更新了注释、body 和 commit message。
- wuxibin89询问是否特定 PyTorch 版本问题，acmore确认是 2x 峰值机制，与版本无关。

- Begunner 修改意见被采纳后，最终 approve。
- 是否是连续内存泄漏 (correctness): 确认是 2x 瞬态峰值，非持续泄漏。
- 特定 PyTorch 版本相关？ (question): 与版本无关，是所有 PyTorch 版本的通用行为。

风险与影响

- 风险:
  1. 回归风险低：变更仅限于 offload_megatron_model_to_cpu 内部，对外无 API 改动，加载路径和 checkpoint 路径已假定 cpu_data 持久存在。
  2. 形状 / 类型不一致静默错误：断言会捕获此类情况，避免静默退化到原 2x 行为，但如果 param_data 在两次卸载间被重建（如模型重初始化），assert 将失败，需要调用方配合重建 cpu_data。
  3. 无测试覆盖：PR body 说明瞬态峰值难以在 CI 中复现，但长期维护需要关注。 - 影响：影响范围：所有使用 Megatron 框架且启用参数 CPU 卸载（param_offload=True）的训练任务。影响程度：严重——直接修复了大规模模型训练的 OOM 崩溃。用户无需修改配置或代码即可获得稳定性提升。 - 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #6095 [fully_async] Fix: fix megatron save and offload in case param_offload is on: 同一文件，修改了 copy_megatron_model_to_cpu / restore_megatron_model_from_cpu，与本 PR 兼容，均依赖 cpu_data 持续存在。
- PR #5751 [megatron, ckpt] fix: reclaim host memory leaked by dist_checkpointing: 同一文件的不同内存问题（glibc anonymous mmap），与本 PR 互补。
- PR #5651 [Megatron] feat: offload optimizer fp32 params to CPU: 同一文件，不同函数（优化器 fp32 offload），无直接重叠。