

PR #7864 完整报告

verl-project/verl

[rollout, perf] fix: avoid full GC before weight resume

合并时间: 2026-09-15 10:20

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7864>

执行摘要

- 一句话: 权重恢复前改用轻量 `empty_cache`, 去掉关键路径 `full GC`
- 推荐动作: 值得精读, 但重点不在代码而在推理链: 改动本身只有一行, 真正的价值是 Issue #7848 + PR body 中“拆分 `empty_cache` / `GC` / `retry` 各自耗时与释放量”的实验方法, 可直接迁移到其他显存清理、profiler、同步点的性能排查。

阅读时建议关注三个设计决策: 一是为什么普通 `empty_cache()` 在这个边界上已经足够 (进程间 allocator handoff 语义 vs 释放活跃张量); 二是为什么 aggressive cleanup 在权重传输后、KV-cache resume 前的那处被保留 (本 PR 的边界克制); 三是无单测情况下的回归防线如何建立——建议后续为 `update_weights()` 的关键路径加一个轻量守卫 (例如断言该边界不再调用 `aggressive_empty_cache`, 或把 GPU A/B 数据纳入 nightly 基准), 否则同类回退仍可能被静默引入。

功能与动机

Issue #7848 记录了 v0.8.0 升级到 v0.9.0 后共置 GRPO 的线上回归: 权重同步中位数从 2.706 s 涨到 5.166 s (+90.9%), 吞吐从 385.68 tok/s 降到 279.49 tok/s (-27.5%)。定位结果显示 v0.9.0 在恢复 rollout 权重前新增了 `aggressive_empty_cache(force_sync=True)`, 移除该调用可省下约 1.133 s (约占版本差距的 46%), 把它加回 v0.8.0 则多花 0.695 s。PR body 明确写道: “controlled A/B tests show its full Python GC and retry materially slow weight sync without a measured GPU-memory benefit”, 并指出该调用由不相关的 DeepSeek-V4 变更 PR #7242 引入且未说明原因。Issue 中的 phase trace 进一步量化: 首次 `empty_cache()` 释放 7.66–40.55 GiB/rank 且仍然必要, 而 GC 释放 0 CUDA 字节却耗时 0.56–0.69 s, 其重试每次触发又消耗 0.55–0.63 s, 显式同步本身不到 0.8 ms。因此目标是在保留显存交接语义的前提下, 移除关键路径上冗余的 GC 与重试。

实现拆解

1. 定位变更入口: 改动落在 `verl/workers/engine_workers.py` 的 `update_weights(self, global_steps=None, mode="auto")`。该函数先解析 `effective_mode`, 当 `effective_mode != "naive"` (分离式 trainer/rollout 走 `checkpoint_engine`) 时直接 return, 因此本次改动只作用于 colocated 同步路径, 异步权重传输路径完全不受影响。
2. 替换关键路径清理调用: 在 `set_expandable_segments(False)` 之后、`log_gpu_memory_usage("Before resume weights")` 之前, 把 `aggressive_empty_cache(force_sync=True)` 换成 `get_torch_device().empty_cache()`。

原因是 PyTorch 与 vLLM 是共置但互相独立的进程，训练侧必须在 vLLM 重新映射权重前把未使用的 reserved allocator block 归还设备；普通 `empty_cache()` 已经完成这次“显存交接”，且不会触碰仍被引用的活跃张量。改用 `get_torch_device()` 而非 `torch.cuda` 也维持了仓库统一的设备抽象，兼容 NPU/AMD 等平台。

3. 保留另一处 aggressive 清理：权重传输 / actor offload 之后、KV-cache resume 之前的 `aggressive_empty_cache` 调用未做改动。也就是说本 PR 是“去掉关键路径上的冗余 GC”，而不是取消 aggressive cleanup，这一取舍在 Issue #7848 中被单独论证过。
4. 后续流程不受影响：`self.rollout.resume(tags=["weights"])` 的分支判定逻辑（SGLang `sleep_level=1` 不 resume、vLLM level-1 sleep 必须 resume）以及 `get_per_tensor_param()` / `update_weights()` 的权重同步序列都原样保留，改动仅在清理语义上收窄。
5. 验证与测试配套：无新增单测。作者说明理由为“a CPU mock would only assert the selected cleanup call”，真正的回归体现在延迟与显存行为上，因此采用 8xH200 受控 A/B 验证：8B dense 4.867 s → 4.207 s (13.6%)、8B parameter offload 3.731 s → 3.031 s (18.8%)、30B-A3B MoE 8.204 s → 7.086 s (13.6%)，六次运行均未 OOM，峰值 allocated/reserved 无有意义差异；完整 pre-commit 套件（Ruff、mypy、生成配置校验、设备 API 校验等）在干净 checkout 下通过。

关键文件：

- `verl/workers/engine_workers.py`（模块 权重同步；类别 source；类型 core-logic；符号 `update_weights`, `aggressive_empty_cache`, `get_torch_device`）：唯一改动文件。
`update_weights()` 在 colocated 权重恢复前的显存清理从 `aggressive_empty_cache(force_sync=True)` 换成 `get_torch_device().empty_cache()`，移除关键路径上的 full Python GC、设备同步与 ≥ 1 GiB 重试；README 级别的“一行改动、关键路径、13%–19% 提速”集中在这里。

关键符号：`update_weights`, `aggressive_empty_cache`, `get_torch_device`

关键源码片段

`verl/workers/engine_workers.py`

唯一改动文件。`update_weights()` 在 colocated 权重恢复前的显存清理从 `aggressive_empty_cache(force_sync=True)` 换成 `get_torch_device().empty_cache()`，移除关键路径上的 full Python GC、设备同步与 ≥ 1 GiB 重试；README 级别的“一行改动、关键路径、13%–19% 提速”集中在这里。

```
async def update_weights(self, global_steps: int = None, mode: str = "auto"):
    # 解析模式: "auto" 回落到 config.rollout.checkpoint_engine.backend;
    # 只有 "naive" (共置、进程内权重同步) 才会走到下面的显存清理代码。
    effective_mode = mode if mode != "auto" else self.config.rollout.checkpoint_engine.backend

    # 0. 非 naive 模式 (分离式 trainer / rollout) 走 checkpoint_engine 并提前 return,
    # 因此本 PR 的改动不会影响 fully_async 等异步权重传输路径。
    if effective_mode != "naive":
        if effective_mode == "delta_sharded":
```

```

        metrics = await self.checkpoint_engine.send_weights(self.actor.engine, global_steps=
        global_steps)
        return metrics or {}
    per_tensor_param, _ = self.actor.engine.get_per_tensor_param()
    metrics = await self.checkpoint_engine.send_weights(per_tensor_param, global_steps=
    global_steps)
    return metrics or {}

set_expandable_segments(False)

# === 本 PR 唯一改动点 ===
# 旧: aggressive_empty_cache(force_sync=True)
# 除 empty_cache 外还执行完整 Python GC、显式设备同步，并在释放 >= 1 GiB 后重试；
# 实测 GC 释放 0 CUDA 字节却多花 0.56-0.69 s，重试再叠加 0.55-0.63 s，
# 而显式同步本身不到 0.8 ms。
# 新：只释放未被引用的 allocator 缓存块，完成 PyTorch 侧到 vLLM 侧的显存交接，
# 不触碰仍然存活的张量；这也是 Issue #7848 中 A/B 提升 13.6%-18.8% 的来源。
get_torch_device().empty_cache()
log_gpu_memory_usage("Before resume weights", logger=logger)

# 1. 恢复 rollout 显存（权重在 sleep 阶段已被释放）
# SGLang 的 sleep_level=1（adapter 模式）从不释放权重，所以不 resume；
# vLLM 的 level-1 sleep 走 CuMemAllocator.sleep(offload_tags=("weights",)),
# 权重确实被 unmap，必须 resume。
is_sglang = self.config.rollout.get("name", "") == "sglang"
if is_sglang:
    resume_weights = self.config.rollout.free_cache_engine and getattr(self.rollout, "sleep_
    level", 2) != 1
else:
    resume_weights = self.config.rollout.free_cache_engine
if resume_weights:
    await self.rollout.resume(tags=["weights"])
log_gpu_memory_usage("After resume weights", logger=logger)

# 2. 后续流程（此处省略）： get_per_tensor_param -> LoRA base sync -> update_weights。
# 权重传输 / actor offload 之后、KV-cache resume 之前的那次 aggressive cleanup 仍然保留，
# 本 PR 只移除权重 resume 之前这一处冗余 GC，而不是取消 aggressive cleanup。

```

评论区精华

本 PR 的讨论量极小：只有一条 Issue 评论和一次无正文的 approve，设计论证几乎全部写在 PR body 与 Issue #7848 里。

- Ibaolin（作者）在 Issue 中说明来源：“Hi @wuxibin89 here is the PR you requested to address <https://github.com/verl-project/verl/issues/7848>”，说明该修复由 reviewer 主动要求提交。
- 作者在 PR body 中预先回应了两个潜在质疑：一是“aggressive cleanup 会不会被一并删掉”，答案是“The existing aggressive-cleanup call remains after weight transfer/offload,

before KV-cache resume”；二是“和 PR #7780 是否重复”，答案是 #7780 改的是 sender-cleanup 边界，两者不重叠。

- 关于测试：作者主动声明“No unit test is added because a CPU mock would only assert the selected cleanup call; the regression is latency and GPU-memory behavior covered by the controlled GPU A/B tests”，用 GPU 实验数据替代单测作为证据。
- wuxibin89 以 APPROVED 状态合入且未留任何 review 评论，说明量化证据说服力足够、无未解决异议；但反过来也意味着“自然 trace 中未出现 GC 有效场景”这一前提没有被第二个人独立复核。
- 关键路径上的 full GC 是否真的必要 (design)：在权重 resume 前只保留普通 empty_cache()，full GC 与重试逻辑从关键路径移除；权重传输后、KV-cache resume 前的 aggressive cleanup 保留。
- 是否补充单元测试 (testing)：以 GPU A/B 实验数据替代 CPU 单测，reviewer 未要求补充；代价是后续若有人重新引入 GC 逻辑，CI 不会拦截。
- 与 PR #7780 的边界是否重叠 (question)：确认两处清理边界互不重叠，本 PR 无需与 #7780 合并或协调。

风险与影响

- 风险：
 1. 显存回归存在条件性：Issue #7848 的合成测试确认“当不可达的 Python 循环引用持有 CUDA 张量时，GC 确实能救回显存”，只是这种条件在自然 trace 中未出现。若某些配置（LoRA adapter、超长序列、MoE 路由缓存等）会制造这类引用环，去掉 GC 后这部分显存不会被归还，可能表现为权重恢复阶段 OOM，而本地 A/B 用的是 Qwen3-8B/30B-A3B 单一模型族，覆盖面有限。
 2. 同步语义依赖需要确认：被移除的 force_sync=True 意味着旧路径在清理后会做一次显式同步，新路径不再强制同步。Issue 测得显式同步本身 <0.8 ms，因此不是性能来源，但若 self.rollout.resume(tags=["weights"]) 暗含对“此前已完成同步”的假设，则跨进程时序保证需要由 vLLM/SGLang 自身提供，这类隐式依赖不会在单测中暴露。
 3. 测试护栏缺失：改动处于每步训练的关键路径 (verl/workers/engine_workers.py:769)，却没有任何自动化回归保护。若后续有人在同一边界重新引入 GC 或重试逻辑，只会有 GPU 侧吞吐回退，CI 不会报错，回退可能再次潜伏到下一个版本（此次回归本身就是由 PR #7242 静默引入的）。
 4. 兼容性风险低：改动使用 get_torch_device() 统一抽象并只影响 effective_mode == "naive" 分支，分离式 checkpoint_engine 路径提前 return，故对 fully_async / delta_sharded 等部署形态无影响。- 影响：对用户：共置 (colocated) 同步 RL 训练的用户直接受益，权重同步中位数下降 13.6%–18.8% (8xH200 上 4.867 s → 4.207 s、3.731 s → 3.031 s、8.204 s → 7.086 s)。Issue #7848 的版本对比显示该调用约占 v0.8.0 → v0.9.0 回归的 46%，因此这是逐步收回 v0.9.0 吞吐损失的重要一步，但并非全部（版本间锁的差异等仍存在）。

对系统：权重同步位于每个训练步的关键路径，延迟降低会在长训练任务上复利放大；显存峰值基本不变 (45.671/45.685GiB、53.121/53.180GiB等对比)，意味着没有用显存换速度。

对团队：本次确立了“关键路径上不做 full Python GC + 重试”的实践基线，并给出了一套可复用的论证范式（phase trace 拆分每次清理的成本、A/B 对比峰值显存）。无 API、无配置、无文档变更，使用与升级成本为零。

- 风险标记：关键路径变更，缺少测试覆盖，条件性显存回归风险，验证依赖 GPU A/B 实验，隐式同步语义依赖

关联脉络

- PR #7851 [worker] fix: sync ranks before resume kv_cache to avoid OOM on non-leader ranks: 改动同一文件 verl/workers/engine_workers.py，且同属 colocated 权重同步 / resume 路径的显存与 rank 协调问题；本 PR 移除的正是该边界上的强制同步 GC，两者需一起理解同步语义与显存释放时序。
- PR #7242 标题未在材料中提供（Issue #7848 指出其为引入 aggressive_empty_cache 的 DeepSeek-V4 变更 PR）：Issue #7848 明确指控“PR #7242 added this call in an unrelated DeepSeek-V4 change without explaining it”，本 PR 实质上是对该次静默引入的性能回退做收口。
- PR #7780 标题未在材料中提供（PR body 描述为配置 separate sender-cleanup 边界的 GC）：同为 GC/ 显存清理边界的调整，作者在 checklist 中专门确认两者作用边界不同、不存在重复修改。
- PR #7770 [training_utils] feat: Add automatic CUDA/NPU OOM snapshots and multi-step memory history: 同属显存行为可观测性建设；本 PR 的结论依赖对“释放多少 GiB / 峰值显存是否变化”的观测，这类内存分析器正是后续守住同类回归的配套工具。