PR #2243 完整报告

THUDM/slime

fix: restore args.ckpt_step after load_other_checkpoint

合并时间：2026-08-12 13:45

原文链接：http://prhub.com.cn/THUDM/slime/pull/2243

执行摘要

- 一句话：修复 load_other_checkpoint 未还原 args.ckpt_step 导致后续加载错迭代
- 推荐动作：建议合入此修复，因为它修复了一个静默的错误权重加载问题，且改动简单、逻辑清晰。值得关注的设计是使用 old_args 元组统一保存和还原多个参数，这是该模式的一次合理扩展，也提示团队在未来修改 load_other_checkpoint 时注意参数生命周期的管理。

功能与动机

PR body 明确指出，--ref-ckpt-step 或 --opd-teacher-ckpt-step 会永久泄漏进 args.ckpt_step，导致同一进程内后续检查点加载解析到错误的迭代。根本原因是 load_other_checkpoint 仅在保存的 old_ckpt_step 非 None 时还原，而 args.ckpt_step 默认是 None，所以正常情况下还原永远不会执行。这会导致 get_load_checkpoint_path_by_args 在每次后续加载（如 --keep-old-actor 时的 old_actor 加载）时使用泄漏的 iteration，静默加载错误权重或直接崩溃。

实现拆解

1. 回归根因定位：slime/backends/megatron_utils/actor.py 的 load_other_checkpoint 中，原本用 old_ckpt_step 临时保存 ckpt_step，但还原操作被错误地放在 if old_ckpt_step is not None 分支内。由于 args.ckpt_step 默认是 None，old_ckpt_step 通常为 None，因此还原逻辑从不执行。
2. 修复方案：将 ckpt_step 加入 old_args 元组（该元组已用于保存和还原 load、no_load_optim、no_load_rng、finetune），并在 load_checkpoint 之后无条件地从 old_args 还原所有字段，包括 ckpt_step。这消除了原代码中“保存值与是否应用覆盖”条件不一致的问题，使还原逻辑在总是发生。
3. 代码结构清理：原有 old_ckpt_step 单独保存和条件还原的代码被删除，统一用 old_args 元组的元组解包赋值，保持了与其他参数还原模式的一致性，简化了控制流并避免未来再次引入类似不一致。

该变更没有修改测试文件，PR body 解释了原因：actor.py 依赖真实 Megatron 运行环境，无法在 CPU 单测中导入；且现有 GPU CI 路径（如 test_qwen3_4B_ppo.py）没有设置 --ref-ckpt-step，因此没有暴露此问题。

关键文件：

- slime/backends/megatron_utils/actor.py（模块 Megatron 工具；类别 source；类型 core-logic）：核心修复文件，load_other_checkpoint 方法调整了参数保存与还原逻辑。

关键符号：load_other_checkpoint

关键源码片段

slime/backends/megatron_utils/actor.py

核心修复文件，load_other_checkpoint 方法调整了参数保存与还原逻辑。

```
def load_other_checkpoint(self, model_tag: str, path: str) -> None:
    # 将 ckpt_step 也纳入 old_args，与其他参数一起统一保存与还原
    old_args = (
        self.args.load,
        self.args.no_load_optim,
        self.args.no_load_rng,
        self.args.finetune,
        self.args.ckpt_step,
    )
    self.args.load = path
    self.args.no_load_optim = True
    self.args.no_load_rng = True
    self.args.finetune = True

    # 仅当显式指定了 ref/teacher 的 ckpt_step 时才覆盖 args.ckpt_step
    if model_tag == "ref" and self.args.ref_ckpt_step is not None:
        self.args.ckpt_step = self.args.ref_ckpt_step
    elif model_tag == "teacher" and self.args.opd_teacher_ckpt_step is not None:
        self.args.ckpt_step = self.args.opd_teacher_ckpt_step

    _, _ = load_checkpoint(
        self.model,
        None,
        None,
        checkpointing_context={},
        skip_load_to_model_and_opt=False,
    )

    # 无条件还原所有 args，包括 ckpt_step，避免泄漏到后续加载
    (
        self.args.load,
        self.args.no_load_optim,
        self.args.no_load_rng,
        self.args.finetune,
        self.args.ckpt_step,
    ) = old_args

    self.weights_backuper.backup(model_tag)
    self._active_model_tag = model_tag
```

评论区精华

该 PR 未被合并，且无 review 评论与讨论。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险是回归风险：在 load_other_checkpoint 被调用的场景（ref 或 teacher 检查点加载）中，此改动改变了 args.ckpt_step 的处理方式。但是修复逻辑与 load 等参数的处理完全一致，且改动极小，风险较低。由于没有相关测试，如果未来有人修改了 load_other_checkpoint 的参数生命周期逻辑，此模式可能再次被破坏。建议在 GPU CI 中增加覆盖 --ref-ckpt-step 的场景，以防回归。
- 影响：影响范围：直接影响 Megatron 后端的多模型检查点加载流程，特别是使用 --ref-ckpt-step 或 --opd-teacher-ckpt-step 的 PPO/OPD 训练场景。修复后，args.ckpt_step 在加载完 ref/teacher 后能正确还原，后续同一进程内的 actor/rollout_actor 权重加载会使用正确的迭代。对用户而言，它修复了潜在的错误权重加载和潜在崩溃；对系统而言，它保证了多阶段训练中检查点加载的确定性。
- 风险标记：缺少测试覆盖

关联脉络

- PR #2213 Fix tau-bench token deltas for reasoning templates: 同为 bugfix，且修改了训练相关逻辑，可能与 ckpt_step 的迭代追踪相关。
- PR #2235 fix: whiten advantages over the DP group that includes context parallel: 同为 Megatron 后端的 bugfix，涉及多进程场景下的正确性问题。