PR #2247 完整报告

THUDM/slime

fix: forward dual-clip PPO epsilon

合并时间：2026-08-12 13:29

原文链接：http://prhub.com.cn/THUDM/slime/pull/2247

执行摘要

- 一句话：转发 eps_clip_c 修复双截断 PPO 未生效
- 推荐动作：值得快速阅读。改动虽小，但补上了配置与损失路径之间的断点；测试用 AST 检查参数透传的做法在缺少直接导出时可借鉴，但也提示此类『接线 bug』最好通过面向行为的集成测试覆盖。

功能与动机

PR body 明确说明：Fix Dual-clip PPO support by forwarding args.eps_clip_c from policy_loss_function to compute_policy_loss. Forward eps_clip_c=args.eps_clip_c in the PPO policy-loss path. 核心问题是 dual-clip PPO 的配置项虽然存在，但未在训练主路径被消费，用户设置 --eps-clip-c 不会影响损失计算。

实现拆解

1. 核心改动：修改 slime/backends/megatron_utils/loss.py 中 policy_loss_function 的 else 分支，将 compute_policy_loss 调用改为显式关键字参数 eps_clip_c=args.eps_clip_c，从而把命令行配置透传到损失计算的裁剪逻辑；CISPO 分支保持调用 compute_cispo_loss 不变。
2. 测试新增：tests/test_policy_loss.py 新增两个 CPU 测试。第一个直接校验 slime/utils/ppo_utils.compute_policy_loss 在负优势下应用 dual-clip 的数值；第二个用 AST 解析 loss.py，断言 policy_loss_function 中存在唯一一次 compute_policy_loss 调用且带 eps_clip_c=args.eps_clip_c 关键字参数，验证训练路径接线。
3. CI 接入：在 .github/workflows/pr-test.yml 与模板 pr-test.yml.j2 的 num_gpus=0 CPU 测试矩阵中加入 test_policy_loss.py，保证新测试在 PR CI 中运行。

关键文件：

- slime/backends/megatron_utils/loss.py（模块 损失计算；类别 source；类型 core-logic；符号 policy_loss_function）：核心修改文件：在 policy_loss_function 中把 args.eps_clip_c 透传给 compute_policy_loss，接通 dual-clip PPO 配置。
- tests/test_policy_loss.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_compute_policy_loss_applies_dual_clip_to_negative_advantages, test_policy_loss_function_forwards_eps_clip_c）：新增测试文件，数值验证 dual-clip 裁剪逻辑，并用 AST 断言训练路径透传参数。

- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：将新增的 test_policy_loss.py 加入 CPU 测试矩阵，确保 PR CI 覆盖该测试。
- .github/workflows/pr-test.yml.j2（模块 CI 配置；类别 infra；类型 infrastructure）：同步测试列表模板，保持生成的工作流与直接维护的 pr-test.yml 一致。

关键符号：policy_loss_function, test_compute_policy_loss_applies_dual_clip_to_negative_advantages, test_policy_loss_function_forwards_eps_clip_c

关键源码片段

slime/backends/megatron_utils/loss.py

核心修改文件：在 policy_loss_function 中把 args.eps_clip_c 透传给 compute_policy_loss，接通 dual-clip PPO 配置。

```
# slime/backends/megatron_utils/loss.py 中 policy_loss_function 的损失计算分支
# 先根据 advantage_estimator 选择损失函数，CISPO 走独立分支，其余走通用 PPO 损失
if args.advantage_estimator == "cispo":
    # CISPO 自身处理 clip 参数，无需透传 eps_clip_c
    pg_loss, pg_clipfrac = compute_cispo_loss(
        ppo_kl, log_probs, advantages, args.eps_clip, args.eps_clip_high
    )
else:
    # 通用 PPO 路径：显式透传 args.eps_clip_c，启用 dual-clip 对负优势的二次裁剪
    pg_loss, pg_clipfrac = compute_policy_loss(
        ppo_kl,
        advantages,
        args.eps_clip,
        args.eps_clip_high,
        eps_clip_c=args.eps_clip_c,
    )
```

tests/test_policy_loss.py

新增测试文件，数值验证 dual-clip 裁剪逻辑，并用 AST 断言训练路径透传参数。

```
# tests/test_policy_loss.py —— CPU 测试，验证 dual-clip PPO 裁剪与其在训练路径的接线

def test_compute_policy_loss_applies_dual_clip_to_negative_advantages():
    ratios = torch.tensor([2.0, 2.0, 0.5])
    ppo_kl = -ratios.log() # 由 ratio 推导的 KL 项
    advantages = torch.tensor([2.0, -2.0, 2.0])

    losses, _ = compute_policy_loss(
        ppo_kl,
        advantages,
        eps_clip=0.2,
        eps_clip_high=0.2,
        eps_clip_c=1.5, # 开启 dual-clip 的下限
    )
```

```
    # 负优势样本（第二个）的损失应被 eps_clip_c 限制为 3.0
    torch.testing.assert_close(losses, torch.tensor([-2.4, 3.0, -1.0]))


def test_policy_loss_function_forwards_eps_clip_c():
    # 用 AST 解析源码，确认 policy_loss_function 向 compute_policy_loss 透传 args.eps_clip_c
    loss_path = Path(__file__).parents[1] / "slime" / "backends" / "megatron_utils" / "loss.py"
    module = ast.parse(loss_path.read_text())
    policy_loss_function = next(
        node for node in module.body
        if isinstance(node, ast.FunctionDef) and node.name == "policy_loss_function"
    )
    compute_policy_loss_calls = [
        node for node in ast.walk(policy_loss_function)
        if isinstance(node, ast.Call) and isinstance(node.func, ast.Name)
        and node.func.id == "compute_policy_loss"
    ]
    assert len(compute_policy_loss_calls) == 1
    eps_clip_c_keyword = next(
        (kw for kw in compute_policy_loss_calls[0].keywords if kw.arg == "eps_clip_c"),
        None,
    )
    assert eps_clip_c_keyword is not None
    assert ast.dump(eps_clip_c_keyword.value) == ast.dump(
        ast.Attribute(value=ast.Name(id="args", ctx=ast.Load()), attr="eps_clip_c", ctx=ast.Load())
    )
```

评论区精华

该 PR 没有任何 review 评论，由维护者 zhuzilin 直接合入；变更简单直接，无公开争议。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。eps_clip_c 是可选参数，若命令行未设置且默认值与 compute_policy_loss 内部缺省一致，则对不使用 dual-clip 的用户无行为变化；但一旦 --eps-clip-c 被设置，负优势分支的损失裁剪会改变，需要回归验证。另外测试中的 AST 断言对源码结构较敏感，未来若将调用改为 functools.partial 或封装函数，该测试可能误报失败而需要同步更新。
- 影响：影响启用 --eps-clip-c 的训练作业，使 dual-clip PPO 的实际行为与文档 / 预期一致；对于未启用该选项的绝大多数用户无影响。团队 CI 增加一个 CPU 测试，几乎无资源压力。
- 风险标记：配置透传可能改变双截断用户训练行为，AST 断言对源码结构敏感

关联脉络

- PR #2235 fix: whiten advantages over the DP group that includes context parallel: 同样修改 slime/backends/megatron_utils/loss.py，属于损失计算模块的并行正确性修复，与

本 PR 在同一函数上下文演进。

- PR #2205 perf: vectorize REINFORCE++ discounted returns: 修改 slime/utils/ppo_utils.py，该文件包含 compute_policy_loss 的实现，与本 PR 共享损失工具链。