PR #2235 完整报告

THUDM/slime

fix: whiten advantages over the DP group that includes context parallel

合并时间：2026-08-12 13:49

原文链接：http://prhub.com.cn/THUDM/slime/pull/2235

执行摘要

- 一句话：修复 CP 下 advantage 归一化错用进程组的问题
- 推荐动作：值得精读。核心改动只有一行进程组参数与一处守卫删除，但 PR body 提供了完整的根因分析、数值验证和死锁论证，展示了分布式训练中集体操作进程组语义的调试方法论。测试设计（gloo 多进程 + 枚举 DP/CP 分解 + 空 mask 场景 + 有界 join）是分布式回归测试的范例，可复用到其他跨 rank 统计逻辑。建议重点关注 loss.py 中 distributed_masked_whiten 调用段及其注释，以及测试对空 mask 场景的处理。

功能与动机

PR body 明确指出：--normalize-advantages 在 context parallelism 下用错误的进程组计算，advantages 按每个 CP rank 的局部统计而非全局统计做 whitening。根因是 compute_advantages_and_returns 从当前 CP rank 的 zigzag 切片构建 all_advs / all_masks，却交给 mpu.get_data_parallel_group()（默认 with_context_parallel=False）做 all-reduce，导致每个 CP rank 对同一条序列的不同部分施加不同的仿射变换，且没有任何 rank 得到全局统计。作者给出的数值示例显示序列两半偏差约 34%，且 slime_validate_args 强制 reinforce_plus_plus 系列使用 --normalize-advantages，CI 中 CP 与 normalize 组合运行未被发现，因为现有测试不检查 whitened 值。

实现拆解

实现分四步：

1. 修正进程组语义：在 slime/backends/megatron_utils/loss.py 的 compute_advantages_and_returns 中，将 mpu.get_data_parallel_group() 改为 mpu.get_data_parallel_group(with_context_parallel=True)。由于 all_advs / all_masks 只包含当前 CP rank 持有的 zigzag 切片，统计量必须在覆盖 CP 维的 DP 组上 reduce。仓库内其他调用点（data.py:186,188、model.py:693、loss.py:1295 传 True；actor.py:106,242 传 False）均显式传参，此处是唯一的隐式调用。
2. 移除 all_masks.numel() > 0 守卫：将 distributed_masked_whiten 调用改为无条件执行。CP rank 在 prompt-heavy 序列下可能合法拥有零个 response token（例如 cp=2, prompt=8000, resp=1000 时响应 token 分布为 [1000, 0]），此时跳过 collective 会让参与 all-reduce 的 rank 集合随数据变化，在组跨越 CP 时必然死锁。distributed_masked_whiten 对空 local tensor 贡献 0，且全局 mask 为空时仍会 raise。

3. 新增多进程回归测试：tests/test_advantage_whiten_cp.py 用 gloo 后端 spawn dp_size * cp_size 个 worker，stub 掉 Megatron 的 mpu 进程组接口，运行真实的 compute_advantages_and_returns，断言同一样本的 whitened advantage 在所有 (dp, cp) 分解下一致且等于单 rank 基线。覆盖 (1,1) (2,1) (1,2) (2,2) (1,4) (4,1)，并特意放入 prompt-heavy 序列让部分 rank 贡献空局部 mask。spawn join 设 180 秒上限，防止回归变成 CI 卡死。
4. CI 注册：在 .github/workflows/pr-test.yml 与模板 .github/workflows/pr-test.yml.j2 的 cpu-unittest 任务中加入 test_advantage_whiten_cp.py（num_gpus: 0）。

关键文件：

- slime/backends/megatron_utils/loss.py（模块 优势计算；类别 source；类型 core-logic；符号 compute_advantages_and_returns）：核心修复文件：compute_advantages_and_returns 中 whitening 的进程组从排除 CP 的 DP 组改为 with_context_parallel=True，并移除 numel() > 0 守卫，彻底解决 CP 下 advantage 统计错误与潜在死锁。
- tests/test_advantage_whiten_cp.py（模块 回归测试；类别 test；类型 test-coverage；符号 _Args, _whiten_worker, cp_slice, _run_case）：新增多进程 CPU 回归测试，用 gloo spawn 验证不同 DP/CP 分解下 whitened advantage 的一致性，覆盖空局部 mask 场景，是本次修复的验证核心。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：在 cpu-unittest job 中注册新增测试，保证 CP whitening 回归测试进入 CI 常驻执行。
- .github/workflows/pr-test.yml.j2（模块 CI 配置；类别 infra；类型 infrastructure）：CI 模板同步注册新测试，保证后续由模板生成的 workflow 也包含该测试。

关键符号：compute_advantages_and_returns, _whiten_worker, cp_slice, _run_case, test_whitened_advantages_are_cp_invariant

关键源码片段

slime/backends/megatron_utils/loss.py

核心修复文件：compute_advantages_and_returns 中 whitening 的进程组从排除 CP 的 DP 组改为 with_context_parallel=True，并移除 numel() > 0 守卫，彻底解决 CP 下 advantage 统计错误与潜在死锁。

```
# all_advs / all_masks 只覆盖本 CP rank 持有的 zigzag 切片，
# 因此统计量必须在“包含上下文并行”的 DP 组上做 all-reduce，
# 否则每个 CP rank 都会用自己切片的均值 / 方差做归一化，
# 同一条序列的两半会被施加不同的仿射变换。
assert (
    all_advs.size() == all_masks.size()
), f"Shape mismatch before whitening: advantages {all_advs.size()}, masks {all_masks.size()
}"


# 该 collective 必须无条件执行：prompt-heavy 序列可能让某个
# CP rank 持有零个 response token，若按 numel() > 0 跳过，
# 会因 rank 参与不一致而卡死 all-reduce。
```

```
        # distributed_masked_whiten 对空 local tensor 贡献 0。
        dp_cp_group = mpu.get_data_parallel_group(with_context_parallel=True)

        whitened_advs_flat = distributed_masked_whiten(
            all_advs,
            all_masks,
            process_group=dp_cp_group,
            shift_mean=True,
        )
        chunk_lengths = [chunk.size(0) for chunk in advantages]
        advantages = list(torch.split(whitened_advs_flat, chunk_lengths))
```

tests/test_advantage_whiten_cp.py

新增多进程 CPU 回归测试，用 gloo spawn 验证不同 DP/CP 分解下 whitened advantage 的一致性，覆盖空局部 mask 场景，是本次修复的验证核心。

```
def cp_slice(x, total_len, response_len):
    """只保留当前 CP rank 拥有的 response 位置（zigzag 切片的两段）。"""
    if cp_size == 1:
        return x
    prompt_len = total_len - response_len
    _, _, _, offsets = get_logits_and_tokens_offset_with_cp(total_len, response_len)
    parts = []
    for start, end in offsets:
        # 全局 offset 先减去 prompt 长度，映射到 response 空间
        lo, hi = max(0, start - prompt_len), max(0, end - prompt_len)
        if hi > lo:
            parts.append(x[lo:hi])
    return torch.cat(parts) if parts else x[:0]


def _whiten_worker(rank, world_size, cp_size, dp_size, master_port, result_dir):
    """单个 spawn 出来的 rank：对本地 CP 切片做 whitening 并导出结果。"""
    cp_rank = rank % cp_size
    dp_rank = rank // cp_size
    stub_megatron_in_worker(cp_size, cp_rank)
    _dist.init_process_group(backend="gloo", rank=rank, world_size=world_size)
    from megatron.core import mpu

    # 无 TP/PP，DP-with-CP 即整个 world；DP-only 组是按 cp_rank 划分的
    # 子集，等价于 Megatron 默认的 get_data_parallel_group()。
    dp_cp_group = _dist.new_group(ranks=list(range(world_size)))
    dp_only_groups = [
        _dist.new_group(ranks=[r for r in range(world_size) if r % cp_size == c])
        for c in range(cp_size)
    ]
    mpu.get_data_parallel_group = lambda with_context_parallel=False, **kw: (
        dp_cp_group if with_context_parallel else dp_only_groups[cp_rank]
    )
```

```
from slime.backends.megatron_utils.loss import compute_advantages_and_returns

# 按 DP rank 轮询分片样本，构造每条序列的 CP 局部 log_probs / masks
my_samples = [i for i in range(len(SEQS)) if i % dp_size == dp_rank]
rollout_data = {
    "log_probs": [cp_slice(torch.zeros(SEQS[i][1]), *SEQS[i]) for i in my_samples],
    "loss_masks": [torch.ones(SEQS[i][1]) for i in my_samples],
    "rewards": [REWARDS[i] for i in my_samples],
    # 其余字段（values / response_lengths 等）用 _Args 默认值补齐
}
compute_advantages_and_returns(_Args(), rollout_data)
# 导出 whitened advantages 供父进程跨分解对比
...
```

评论区精华

PR 没有 review 评论。唯一讨论是作者 keepkeen 在 Issue 评论中请维护者 zhuzilin review，说明该修复针对 CP 下 advantage whitening 的可复现训练正确性问题，并新增多进程 CPU 回归测试覆盖不同 DP/CP 分解与空 mask 场景，GitHub 报告可合并且 CI 全绿；PR 随后由 zhuzilin 合并。PR body 中的技术论证本身值得重视：作者用数值示例证明修复前后序列两半的 whitened advantage 从 +1.707037 / +1.273883 收敛到一致的 +1.439168，并解释了 rank 依赖的 collective 守卫为何必然导致死锁——这是对分布式集体操作数据依赖风险的清晰剖析。

- 请求 review 与测试覆盖确认 (question): PR 随后由 zhuzilin 合并，未产生进一步 review 讨论。

风险与影响

- 风险：
  1. 核心训练路径变更：compute_advantages_and_returns 是所有 advantage estimator（grpo/gspo/cispo/ppo/reinforce_plus_plus 等）的公共路径，修改会影响全部使用 --normalize-advantages 的训练，不只是 CP 场景；需要确认 with_context_parallel=True 在 cp_size=1 时与默认组等价（Megatron 语义下成立）。
  2. collective 空张量语义依赖：移除守卫后，无条件调用 distributed_masked_whiten 的正确性依赖其“空 local tensor 贡献 0、全局为空时 raise”的实现，PR 未直接修改或验证该函数，建议确认其空张量路径。
  3. 测试覆盖局限：新增测试基于 gloo CPU 多进程，不覆盖真实 GPU/NCCL 路径；现有 GPU CI（如 test_qwen3_4B_ppo.py 的 CP 组合）不检查 whitened 值，因此 GPU 上仍需人工验证。
  4. 行为变化：原来 numel()==0 时静默跳过 whitening，现在会无条件参与；若全局 mask 为空会在 distributed_masked_whiten 内显式报错，属于把潜在静默错误变成显式失败，方向上更安全。 - 影响：影响所有结合 context parallelism 与 --normalize-advantages 的训练运行，包括 reinforce_plus_plus / reinforce_plus_plus_baseline（被 slime_validate_args 强制要求 normalize）以及 CI

中已有的 CP 组合用例。修复前 advantage 统计偏差可达约 34%，会直接扭曲策略梯度信号、损害收敛质量；修复后不同 DP/CP 分解下的 whitened advantage 与单 rank 基线一致，训练结果可复现。对团队而言，需要重跑相关 CP 训练以验证效果，同时新增的 CPU 回归测试为后续进程组语义变更提供了防线。 - 风险标记：核心训练路径变更，collective 空张量语义依赖，GPU/NCCL 路径未覆盖，训练正确性修复

关联脉络

- PR #2208 Support reloading the default process group: 同一进程组语义与生命周期基础设施：该 PR 新增可重载的默认进程组（reloadable_process_group），同样改动 actor.py / rollout.py 与 pr-test.yml，与本 PR 对 DP/CP 进程组语义的修正同属分布式训练集体操作正确性范畴。
- PR #2213 Fix tau-bench token deltas for reasoning templates: 同为训练正确性修复，涉及 token 掩码与优势相关计算的精细语义；且 CI 中新增的 test_advantage_whiten_cp.py 与既有 test_loss_cp_invariance.py 相邻，说明 CP 相关正确性测试正在形成系列。