PR #7340 完整报告

verl-project/verl

[megatron] fix: bugfix qwen 3 qwen 3.5 router replay

合并时间：2026-08-12 20:41

原文链接：http://prhub.com.cn/verl-project/verl/pull/7340

执行摘要

- 一句话：修复 Qwen3.5 Megatron router replay 静默失效
- 推荐动作：值得精读。该 PR 是“进程级全局可变列表在复杂构建流程下寻址失效”的典型案例，展示了从“按列表位置”到“按实际对象树”寻址的设计范式；MTP layer_number alias 的二次修复说明 review 对边界条件的价值。阅读时可重点看 iter_model_routers 的 decoder scope 与 set_router_replay_data 的 MoE 序号换算逻辑，并关注后续 VPP 场景下 layer_number 语义是否需要在真实训练中验证。

功能与动机

PR body 明确指出 Qwen3VLGPTModel.__init__ 在 megatron bridge 中会构建两次 decoder，被丢弃的 routers 已 append 进全局列表且从不移除，导致 len(router_instances) 是实际层数的两倍（Qwen3.5-35B-A3B 为 80/40，Qwen3.5-122B-A10B 为 48/24），而 slice 从 offset 0 开始恰好取到孤儿 router，造成“静默”破坏：get_replay_topk 走 default 路径、backward 从不进入 REPLAY_BACKWARD，没有 error 也没有 warning。旧 PR #4567 报告过相同症状但未修复合并。

实现拆解

1. 定位根因：在 verl/utils/megatron/router_replay_utils.py 中确认 set_router_replay_data 原先通过 RouterReplayHelper.get_micro_batch_router_list 取全局列表并做位置切片，无法应对全局列表多于实际层的场景。
2. 新增模型树寻址：新增 iter_model_routers(model)，支持单个 module 或 VPP chunk 列表，遍历 chunk.decoder 子树（第二个 commit 将遍历范围 scope 到 decoder），筛出带 router_replay 且 layer_number 非空的 TopKRouter，按模块自身的 1-based 层号 yield。
3. 改造 set_router_replay_data：增加可选参数 model=None，传入时改用 iter_model_routers 逐层写 set_target_indices，并保留原有 index_by_layer / MoE 序号换算逻辑；model=None 时退回原有位置切片路径，保证向后兼容。
4. 新增 action 切换：新增 set_model_router_replay_action(model, action)，在 verl/workers/engine/megatron/transformer_impl.py 的 forward_step 中三处调用：replay forward 前切 REPLAY_FORWARD、写 targets 时传 model=unwrapped_model、forward 后切 REPLAY_BACKWARD。

5. 测试配套：新增 tests/utils/megatron/test_router_replay_model_walk_on_cpu.py，通过 MagicMock stub 掉 megatron-core 依赖，用 FakeTopKRouter 构造 orphans + model 场景，验证 targets/action 只落到模型自身 router、孤儿不受影响、MTP 层不被寻址。

关键文件：

- verl/utils/megatron/router_replay_utils.py（模块 路由重放；类别 source；类型 core-logic；符号 set_router_replay_data, iter_model_routers, set_model_router_replay_action）：核心修复所在：新增 iter_model_routers 与 set_model_router_replay_action，并给 set_router_replay_data 增加 model 参数，从全局列表位置寻址改为按模型对象树寻址。
- verl/workers/engine/megatron/transformer_impl.py（模块 引擎；类别 source；类型 entrypoint；符号 forward_step）：forward_step 作为入口，把 unwrapped_model 传入 set_router_replay_data，并在两个 action 切换循环后调用 set_model_router_replay_action，是修复实际生效的调用点。
- tests/utils/megatron/test_router_replay_model_walk_on_cpu.py（模块 路由重放；类别 test；类型 test-coverage；符号 FakeTopKRouter, _build_model, _routers, orphans_then_model）：新增 CPU 单测，精确复现孤儿 router 场景并验证 targets/action 落到模型自身、MTP 不被寻址，是防止回归的关键保障。

关键符号：iter_model_routers, set_model_router_replay_action, set_router_replay_data, forward_step

关键源码片段

verl/utils/megatron/router_replay_utils.py

核心修复所在：新增 iter_model_routers 与 set_model_router_replay_action，并给 set_router_replay_data 增加 model 参数，从全局列表位置寻址改为按模型对象树寻址。

```
# 遍历实际参与 forward 的模型，按模块自身的 1-based layer_number 寻址 router。
# 背景：RouterReplay 会把自己 append 到全局 RouterReplay.router_instances，
# 而 Qwen3VLGPTModel 构建时曾两次创建 decoder，被丢弃 decoder 里的 router
# 残留成孤儿（2N 条），若按全局列表位置切片会瞄准错误的对象。
def iter_model_routers(model):
    # model 可能是单个 module，也可能是 VPP 的多个 chunk
    for chunk in model if isinstance(model, list | tuple) else [model]:
        # 只遍历 decoder 子树：MTP 层的 layer_number 从 1 重新计数，
        # 会与 decoder 第 1 层 alias，而 replay 张量里没有 MTP 行。
        # GPTModel 和 HybridModel 中 mtp 都是 decoder 的兄弟节点，
        # scope 到 decoder 可天然排除 MTP。
        for module in getattr(chunk, "decoder", chunk).modules():
            router = getattr(module, "router_replay", None)
            if isinstance(module, TopKRouter) and router is not None and module.layer_number is
            not None:
                yield module.layer_number, router


def set_model_router_replay_action(model, router_replay_action):
    # 与 set_router_replay_data 的 walk 保持一致，把 REPLAY_FORWARD / REPLAY_BACKWARD
```

```
# 切到真正会 forward 的 router 上，避免切到孤儿导致 backward 读取陈旧状态。
for _, router in iter_model_routers(model):
    router.set_router_replay_action(router_replay_action)

    # 有 model 时基于模型自身的 router 写 targets，等价于“按对象寻址”；
    # 无 model 时退回原来的全局列表位置切片，保证向后兼容。
    if model is not None:
        for layer_number, router in iter_model_routers(model):
            layer_idx = layer_number - 1
            # R2 且混合 dense/MoE 时，layers_topk_idx_reshape 仅含 MoE 层，
            # 需要换算成 MoE 层序号；R3 或全 MoE 模型则直接用层号。
            idx = layer_idx if index_by_layer else sum(
                1 for i in range(layer_idx) if is_moe_layer(tf_config, i)
            )
            if 0 <= idx < layers_topk_idx_reshape.shape[0]:
                router.set_target_indices(
                    layers_topk_idx_reshape[idx].to(torch.int64),
                    replay_mask=replay_mask_rmpad_split,
                )
        return
```

verl/workers/engine/megatron/transformer_impl.py

forward_step 作为入口，把 unwrapped_model 传入 set_router_replay_data，并在两个 action 切换循环后调用 set_model_router_replay_action，是修复实际生效的调用点。

```
    # 进入 replay forward 前，先强制把 action 切到模型真实 router 上
    if RouterReplayHelper.is_replay_backward_action(self.tf_config, vp_rank):
        router_instance_list = RouterReplayHelper.get_micro_batch_router_list(self.tf_config,
        vp_rank)
        for router in router_instance_list:
            router.set_router_replay_action(RouterReplayAction.REPLAY_FORWARD)
        set_model_router_replay_action(unwrapped_model, RouterReplayAction.REPLAY_
        FORWARD)

    # 写 targets 时把未包装的模型传进去，按模型对象树寻址而非全局列表
    if RouterReplayHelper.is_replay_forward_action(self.tf_config, vp_rank):
        layers_topk_idx = model_inputs["routed_experts"]
        replay_mask = None
        if self.engine_config.router_replay.mode == "R3":
            layers_topk_idx = align_r3_router_replay_data(layers_topk_idx, input_ids)
            replay_mask = build_r3_replay_mask(input_ids, batch["response_mask"])
        set_router_replay_data(
            layers_topk_idx,
            None,
            self.tf_config,
            vp_rank,
            replay_mask=replay_mask,
            local_cp_size=local_cp_size,
            model=unwrapped_model,
```

```
        )

    # forward 结束后切到 REPLAY_BACKWARD，同样作用于模型自身 router
    if RouterReplayHelper.is_replay_forward_action(self.tf_config, vp_rank):
        router_instance_list = RouterReplayHelper.get_micro_batch_router_list(self.tf_config,
        vp_rank)
        for router in router_instance_list:
            router.set_router_replay_action(RouterReplayAction.REPLAY_BACKWARD)
        set_model_router_replay_action(unwrapped_model, RouterReplayAction.REPLAY_
        BACKWARD)
```

评论区精华

核心讨论围绕 iter_model_routers 的遍历范围展开：wuxibin89 在 line 502 提出“MTP 层的 layer_number 也是 1-based，是否会破坏 MTP + router replay”；HollowMan6 回应当前 router replay 并不覆盖 MTP（get_moe_num_layers_to_build 不计入 mtp_num_layers），但未来引入 MTP 时需小心；EricMarcus-ai 承认初版遍历 chunk.modules() 会包含 MTP，随即改为 scope 到 decoder 并新增 test_mtp_routers_are_not_addressed，同时说明 R3 模式下 MTP router replay 本身定义不清。最终 HollowMan6 批准并评价“LGTM, thanks!”。

- iter_model_routers 是否会误伤 MTP 层？ (correctness): 将 iter_model_routers 的遍历范围限定在 chunk.decoder 子树，MTP 层不再被寻址，行为与原先 positional 路径一致。

风险与影响

- 风险：
  1. VPP 层号语义不确定性：iter_model_routers 依赖 module.layer_number，若 VPP 场景下该属性为全局层号而非局部层号，layer_number - 1 可能越界；虽有 0 <= idx < shape[0] 保护，但越界时会静默跳过目标写入，需要在实际 VPP 训练中验证。
  2. Megatron-Core 符号依赖：router_replay_utils.py 新增了 from megatron.core.transformer.moe.router import TopKRouter 的硬导入，若 megatron-core 版本变化导致该模块路径变更，会直接 import 失败（仅在启用 router replay 的路径上 import，普通训练不受影响）。
  3. 行为兼容性：set_router_replay_data 的 model 分支会在写入后直接 return，绕过了原有 get_current_rank_layer_info 的位置推导；只要调用方（transformer_impl）传了 model，旧逻辑不再执行，但保留 model=None 作为 fallback，降低回归面。
  4. 性能开销：每个 microbatch 都会遍历模型模块树，作者自测为若干微秒，风险低；但若未来模型规模进一步增大或 VPP chunk 数量变多，可考虑 memoization。 - 影响：用户侧：启用 Qwen3-VL / Qwen3.5 的 Megatron router replay（R2/R3）训练不再静默失效，MoE 路由与 rollout 记录对齐，训练正确性恢复；其他模型由于没有孤儿 router，行为与原先完全一致。系统侧：forward_step 每微批多一次模型树遍历，开销微秒级；改动集中在 router replay 工具函数与 Megatron worker 入口，代码量小（+189/-4）。团队侧：补上了 CPU 单测，彻底覆盖了 #4567 遗留问题，并为后续引入 MTP router replay 划清了边界。 - 风险标记：核心训练路径改动，静默失效类缺陷修复，VPP 下层号语义待验证，MTP 边界

关联脉络

- PR #4567 [megatron] fix: Fix RouterReplay/TopKRouter mismatch to support qwen3vl: 同一症状的历史 PR：Qwen3-VL 下 RouterReplay.router_instances 与 TopKRouter 不匹配；作者在 body 中明确引用，旧 PR 未修复未合并。
- PR #6555 [megatron] feat: add dynamic context parallel scheduling: 同文件 verl/utils/megatron/router_replay_utils.py 的大改动，引入 R3 replay mask、动态 CP 等；本 PR 的 model walk 寻址与 R2/R3 的 index_by_layer 逻辑在同一文件内协作。
- PR #7297 [megatron] fix: make DeepSeek-V4 context parallelism actually runnable: 同样修改 verl/utils/megatron/router_replay_utils.py，传递 CP 布局；说明该文件是 Megatron 多特性共享核心，后续改动需注意交互。