PR #2185 完整报告

THUDM/slime

Support routed_experts_start_len

合并时间：2026-07-07 16:14

原文链接：http://prhub.com.cn/THUDM/slime/pull/2185

执行摘要

- 一句话：支持 routed_experts_start_len 处理部分专家路由
- 推荐动作：此 PR 改动虽小，但涉及对 routed experts 数据拼接逻辑的扩展，对于理解分块解码场景下的数据流有帮助。建议阅读 slime/utils/types.py 中的新增拼接逻辑；actor.py 中的单行改动可视为对 routing replay 的完善。

功能与动机

为了支持分块解码场景（如 speculative decoding 或分步 rollout），引擎可能分多次发送 routed experts 元数据，需要根据 routed_experts_start_len 将新接收的 experts 信息拼接到已有数据后部。

实现拆解

1. 数据引入与校验：在 slime/utils/types.py 的 _apply_meta_info 方法中，从 meta_info 读取 routed_experts_start_len（默认为 0），并校验其为非负整数。
2. 行数计算调整：将 expected_rows 的计算由 len(self.tokens) - 1 改为 max(0, len(self.tokens) - 1 - routed_experts_start_len)，表明新接收的 routed experts 只对应从 routed_experts_start_len 开始的后缀 tokens。
3. 数据拼接逻辑：当 routed_experts_start_len == 0 时直接覆盖 self.rollout_routed_experts；否则从已有的 self.rollout_routed_experts 中取前 routed_experts_start_len 行，与新数据拼接，并补充类型检查与异常处理。
4. 条件放宽：在 slime/backends/megatron_utils/actor.py 中，can_reuse_log_probs_in_loss 条件中原本为 not self.args.use_routing_replay，现改为 not self.args.use_routing_replay or self.args.use_rollout_routing_replay，允许在启用 rollout routing replay 时复用 log probs。

关键文件：

- slime/utils/types.py（模块 数据模型；类别 source；类型 core-logic；符号 _apply_meta_info）：核心数据模型变更，添加 routed_experts_start_len 解析与拼接逻辑。
- slime/backends/megatron_utils/actor.py（模块 Actor；类别 source；类型 core-logic；符号 train_actor）：调整 can_reuse_log_probs_in_loss 条件，兼容 routing replay 场景。

关键符号：_apply_meta_info, train_actor

关键源码片段

slime/utils/types.py

核心数据模型变更，添加 routed_experts_start_len 解析与拼接逻辑。

```
# slime/utils/types.py 中 _apply_meta_info 方法片段
routed_experts = decode_int32_meta_array(meta_info, "routed_experts")
if routed_experts is not None:
    if args is None:
        raise ValueError("args is required to decode routed experts metadata.")
    # 新增：从 meta_info 中读取 routed_experts_start_len，默认为 0
    routed_experts_start_len = int(meta_info.get("routed_experts_start_len", 0) or 0)
    if routed_experts_start_len < 0:
        raise ValueError(
            f"SGLang routed_experts_start_len must be non-negative, got {routed_experts_start_len}
            ."
        )
    # 计算新数据应覆盖的 token 行数（排除 start_len 之前的旧行）
    expected_rows = max(0, len(self.tokens) - 1 - routed_experts_start_len)
    expected_numel = expected_rows * args.num_layers * args.moe_router_topk
    if routed_experts.numel() != expected_numel:
        raise ValueError(
            "SGLang routed_experts element count does not match sample tokens: "
            f"got={routed_experts.numel()}, expected={expected_numel} "
            f"(tokens={len(self.tokens)}, routed_experts_start_len={routed_experts_start_len}, "
            f"num_layers={args.num_layers}, moe_router_topk={args.moe_router_topk})."
        )
    # 先 reshape
    routed_experts = routed_experts.reshape(
        expected_rows, args.num_layers, args.moe_router_topk,
    )
    if routed_experts_start_len == 0:
        # 无偏移时直接覆盖
        self.rollout_routed_experts = routed_experts
    else:
        # 有偏移时：取已有数据的前 start_len 行，与新的 routed_experts 拼接
        existing = self.rollout_routed_experts
        if existing is None:
            raise ValueError(
                "Cannot append partial routed experts without existing routed experts "
                f"(routed_experts_start_len={routed_experts_start_len})."
            )
        if not torch.is_tensor(existing):
            existing = torch.as_tensor(existing, dtype=routed_experts.dtype)
        if existing.shape[0] < routed_experts_start_len:
            raise ValueError(
                "Existing routed experts shorter than routed_experts_start_len: "
                f"existing_rows={existing.shape[0]}, routed_experts_start_len={routed_experts_start_
                len}."
```

```
        )
    self.rollout_routed_experts = torch.cat(
        [existing[:routed_experts_start_len], routed_experts],
        dim=0,
    )
```

评论区精华

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 数据一致性：如果 routed_experts_start_len 与上游引擎发送的 experts 行数不匹配（如 existing.shape[0] < routed_experts_start_len），会触发 ValueError，属于防御性设计，但可能导致训练中断。
  2. 性能影响：当 routed_experts_start_len > 0 时，每次调用 _apply_meta_info 都会执行 torch.cat，若频繁小批量调用可能增加内存分配开销。
  3. 无测试配套：本次改动未包含测试文件，可能遗漏了边界条件（如 routed_experts_start_len 恰好等于现有行数、existing 为空等）的验证。 - 影响：影响范围：仅影响使用 routed experts 的 MoE 模型在分块解码场景下的数据拼接行为，正常解码（routed_experts_start_len=0）无变化。can_reuse_log_probs_in_loss 的放宽可提升部分场景的性能。影响程度：中等。核心数据路径变更（Sample._apply_meta_info）可能影响所有 rollout 样本处理，但非默认路径。
- 风险标记：缺少测试覆盖，核心数据路径变更

关联脉络

- PR #2175 Fix R3 for allgather_cp: 涉及 routing replay 修复，与本 PR 对 can_reuse_log_probs_in_loss 的调整相关。
- PR #2181 [3/n] Disaggregated rollout: engine-side /pull_weights: 同为分布式 rollout 基础设施改进，可能共用 routed experts 数据流。