

PR #7075 完整报告

verl-project/verl

[trainer] fix: exclude the never-routed final response token from rollout MoE load-balance metrics

合并时间: 2026-07-16 22:29

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7075>

执行摘要

- 一句话: 修复 MoE 指标中最后一个 token 的幻影计数
- 推荐动作: 建议精读 `_compute_rollout_moe_load_counts` 中排除最后一个有效位置的向量化实现, 以及应对空边界和负值填充的稳健设计。合并后应监控指标变化是否符合预期, 避免阈值告警。该 PR 是 MoE 负载均衡指标正确性的关键修复。

功能与动机

来自 SGLang 的 `routed_experts_capturer.py` 每个请求只返回 `seqlen - 1` 条记录 (参见 sglang issue #20964), 因为最后一个 token 没有 MoE 路由记录。但在 verl 的 `AgentLoopWorker._postprocess` 中, 组装缓冲区被零填充, 使最后一个位置出现幻影专家 0 分配。这导致指标高估专家 0 负载, 在 128 专家、2k 响应长度时偏差约 6%, 512 长度时达 25%, 且无法平均消除。需要修复让指标反映真实负载。

实现拆解

1. 边界检查与早期返回: 在 `_compute_rollout_moe_load_counts` 的输入验证部分增加 `response_len == 0` 的检查, 避免后续向量化操作在空维度上失败, 并消除 `routed_experts[:, -0:]` 的危险切片行为。
2. 定位每个序列的最后一个有效 token: 利用 `torch.arange` 生成位置序列, 配合 `torch.where` 仅在有效掩码位置保留索引, 否则置为 -1, 然后 `amax` 得到每行的最后一个有效位置。
3. 构建排除掩码: 将 `response_mask` 中对应最后有效位置的位置置为 False, 通过广播比较生成布尔张量, 避免使用 `nonzero()` 引起的 host-device 同步。
4. 选择有效路由数据: 使用更新后的掩码索引 `routed_experts`, 被排除的 token (包含最后一个有效响应位置) 不再参与后续计数。
5. 统计专家分配: 保持原有的 `bincount` 统计逻辑, 按层计算专家计数。

此外, 测试文件新增 5 个测试函数: 验证填充值无影响、负值容错、不等长序列独立移除、单 token 响应、以及零长度掩码场景。

关键文件:

- `verl/trainer/ppo/metric_utils.py` (模块 训练器; 类别 source; 类型 core-logic; 符号 `_compute_rollout_moe_load_counts`): 核心逻辑修改文件, 在

`_compute_rollout_moe_load_counts` 中添加了排除最后一个有效响应位置的逻辑，并增加了边界保护。

- `tests/trainer/ppo/test_rollout_moe_lb_metrics_on_cpu.py` (模块 测试; 类别 `test`; 类型 `test-coverage`; 符号 `test_rollout_moe_load_balance_metrics_last_token_filler_is_ignored`, `test_rollout_moe_load_balance_metrics_survive_negative_filler`, `test_rollout_moe_load_balance_metrics_drop_last_valid_per_sequence`, `test_rollout_moe_load_balance_metrics_single_token_response_yields_nothing`) : 新增和修改了 5 个测试函数, 覆盖填充值无关性、负值容错、不等长序列独立移除、单 token 响应及零长度掩码场景, 确保修复正确且边界安全。

关键符号: `_compute_rollout_moe_load_counts`

关键源码片段

`verl/trainer/ppo/metric_utils.py`

核心逻辑修改文件, 在 `_compute_rollout_moe_load_counts` 中添加了排除最后一个有效响应位置的逻辑, 并增加了边界保护。

```
def _compute_rollout_moe_load_counts(
    routed_experts: torch.Tensor | None,
    response_mask: torch.Tensor | None,
    num_experts: int | None,
) -> torch.Tensor | None:
    """Count routed experts in response tokens as [num_layers, num_experts].

    Each sequence's last valid response position is excluded: that token is
    sampled but never fed back through the model, so it carries no routing
    record (its slot is filler, not data).
    """
    if routed_experts is None or response_mask is None or num_experts is None or num_experts
    <= 0:
        return None
    if routed_experts.dim() != 4:
        logger.warning("Expected routed_experts with shape [bsz, seqlen, layers, topk], got %s",
            routed_experts.shape)
        return None
    if response_mask.dim() != 2 or response_mask.shape[0] != routed_experts.shape[0]:
        logger.warning(
            "Response mask shape %s is incompatible with routed_experts %s", response_mask.
            shape, routed_experts.shape
        )
        return None

    response_len = response_mask.shape[1]
    if response_len == 0:
        return None
    if routed_experts.shape[1] < response_len:
        logger.warning(
```

```

        "routed_experts sequence length %s is shorter than response length %s",
        routed_experts.shape[1],
        response_len,
    )
    return None

response_routed_experts = routed_experts[:, -response_len:]
response_mask = response_mask.to(device=response_routed_experts.device, dtype=torch.
bool)
# 最后一个响应 token 被采样但未参与前向传播，所以没有路由记录；
# 它的槽位由批处理组装填充（零填充，参见 AgentLoopWorker._postprocess）。
# 如果将其计入，每个序列会给专家 0 贡献 num_layers * topk 个幻影分配，因此需要排除。
positions = torch.arange(response_len, device=response_mask.device)
last_valid = torch.where(response_mask, positions, positions.new_full((), -1)).amax(dim=-1)
has_valid = last_valid >= 0
is_last_valid = (positions.unsqueeze(0) == last_valid.unsqueeze(1)) & has_valid.unsqueeze(1)
response_mask = response_mask & ~is_last_valid
selected = response_routed_experts[response_mask]
selected = selected.detach().to(device="cpu", dtype=torch.long)
if selected.numel() == 0:
    return None
if selected.min() < 0 or selected.max() >= num_experts:
    logger.warning(
        "Skipping rollout MoE load-balance metrics because routed expert ids are outside [0,
        %s): min=%s max=%s",
        num_experts,
        selected.min().item(),
        selected.max().item(),
    )
    return None

# selected: [num_response_tokens, num_layers, topk]. Count every top-k slot
# without materializing a large [tokens, layers, topk, experts] one-hot tensor.
return torch.stack(
    [
        torch.bincount(selected[:, layer_idx, :].flatten(), minlength=num_experts)
        for layer_idx in range(selected.shape[1])
    ]
)

```

评论区精华

- gemini-code-assist[bot]指出两个问题：空响应长度时 `amax` 会崩溃且 `routed_experts[:, -0:]` 错误地返回整个张量；`nonzero()` 引起 host-device 同步开销，建议向量化。
- zsnoob采纳建议：添加 `response_len == 0` 早期返回并增加回归测试，用广播比较掩码替换 `nonzero()` 实现。
- 空响应长度处理 (correctness): zsnoob 添加了 `response_len == 0` 的早期返回，并增加了回归测试。

- nonzero host-device 同步 (performance): zsnoob 采纳了建议，用广播比较掩码实现，并移除了显式 clone()。

风险与影响

- 风险：
 1. 边界条件：若 response_mask 无有效位置（全 False），amax 在 -1 上操作仍可工作，但有 has_valid 的保护，掩码不会错误排除；但需保证输入不出现全 False 掩码（由上游保证）。
 2. 向后兼容：指标键名与形状不变，但实际计数值下降（排除幻影计数）。依赖绝对值或历史基线的监报告警需调整阈值，用户可能观察到专家 0 负载突然下降。
 3. 性能：新增操作为全向量化，额外开销极低，但未在大规模 batch 上压测；理论上不会成为瓶颈。
- 影响：
 - 用户：所有启用 rollout.moe_load_balance_metrics_interval 的用户将获得修正后的 MoE 负载均衡指标，专家 0 的虚高负载消失，负载均衡决策更准确。
 - 系统：负值填充不再导致整个指标静默失效，提高了监控可靠性。与重放端（#6473）的行为对齐，形成完整一致的数据流。
 - 团队：测试覆盖率提升，边界情况得到保护，未来重构更有信心。
 - 风险标记：边界条件处理，核心指标修正，向后兼容需注意

关联脉络

- PR #6853 [trainer, rollout] feat: log rollout moe load-balance metrics: 该 PR 引入了原始 rollout MoE 负载均衡指标，本 PR 修复了其中的系统性偏差。
- PR #6473 [megatron] feat: support DeepSeek V4 GRPO: 本 PR 是 build_r3_replay_mask 的指标端对应物，后者在重放端跳过相同行。