PR #6464 完整报告

verl-project/verl

[megatron] fix: clamp num_tokens=0 in MTP loss & add normalized scale for MTP per token loss

合并时间：2026-06-05 15:44

原文链接：http://prhub.com.cn/verl-project/verl/pull/6464

执行摘要

- 一句话：修复 MTP 损失中 num_tokens=0 导致的 NaN 与梯度归一化
- 推荐动作：建议精读，特别是 per-token 损失梯度归一化的设计：MTP 滚动后 token 数减少，需要重新缩放以对齐主损失的 per-token 梯度。这是 Megatron 训练中易被忽视的细节。

功能与动机

MTP 损失在上下文并行场景下，当整个 CP 组被 mask 时 num_tokens=0，导致损失计算中出现 NaN。此外，上游 Megatron-LM 已修正 per-token 损失的梯度缩放，需要同步。PR body 明确引用了上游修复：https://github.com/NVIDIA/Megatron-LM/pull/3396 和 https://github.com/NVIDIA/Megatron-LM/pull/3159。

实现拆解

1. 记录原始 token 数：在滚动 labels 之前，记录 original_num_tokens = loss_mask.sum()，用于后续重新缩放 MTP 梯度。
2. 安全化日志损失计算：将 torch.sum(mtp_loss) / num_tokens 替换为条件判断，当 num_tokens > 0 时正常计算，否则返回零张量，避免 NaN 传播到 tracker。
3. 修正 per-token 损失梯度缩放：当 calculate_per_token_loss=True 时，MTP 的 MTPLossAutoScaler 的梯度因子从 mtp_loss_scale * mtp_loss 改为 mtp_loss_scale * mtp_loss * (original_num_tokens / num_tokens_safe)，这样最终梯度除以 total_num_tokens（主损失 token 数）后仍保持正确的 per-token 梯度。分母 num_tokens_safe 通过 torch.clamp(num_tokens, min=1) 防止除零。
4. 安全化非 per-token 模式：在 calculate_per_token_loss=False 的 else 分支，也将 num_tokens clamp 到 1。

关键文件：

- verl/models/mcore/mtp_patch.py（模块 模型；类别 source；类型 data-contract）：MTP 损失计算核心逻辑，修复 num_tokens=0 导致的 NaN 并引入正确的梯度归一化

关键符号：未识别

关键源码片段

verl/models/mcore/mtp_patch.py

MTP 损失计算核心逻辑，修复 num_tokens=0 导致的 NaN 并引入正确的梯度归一化

```
# verl/models/mcore/mtp_patch.py 关键片段：损失计算与梯度归一化

# 在滚动 labels 之前记录原始 token 数
original_num_tokens = loss_mask.sum()

for mtp_layer_number in range(self.config.mtp_num_layers):
    mtp_labels, _ = roll_tensor(
        mtp_labels, shifts=-1, dims=-1,
        cp_group=cp_group, packed_seq_params=packed_seq_params,
    )
    loss_mask, num_tokens = roll_tensor(
        loss_mask, shifts=-1, dims=-1,
        cp_group=cp_group, packed_seq_params=packed_seq_params,
    )
    # ... 计算 mtp_loss 和 mtp_loss_scale ...

    # 安全计算日志损失，避免 num_tokens=0 时 NaN
    mtp_loss_for_log = (
        torch.sum(mtp_loss) / num_tokens
        if num_tokens > 0 else mtp_loss.new_tensor(0.0)
    )

    if self.config.calculate_per_token_loss:
        # 当 calculate_per_token_loss 启用时，finalize_model_grads 会
        # 将所有梯度除以 total_num_tokens（来自主损失）。但由于 MTP
        # 滚动后有效 token 数减少，需要重新缩放以保证正确的 per-token 梯度权重。
        num_tokens_safe = torch.clamp(num_tokens, min=1)
        hidden_states = MTPLossAutoScaler.apply(
            hidden_states,
            mtp_loss_scale * mtp_loss * (original_num_tokens / num_tokens_safe),
        )
    else:
        safe_num_tokens = num_tokens.clamp(min=1)
        hidden_states = MTPLossAutoScaler.apply(
            hidden_states, mtp_loss_scale * mtp_loss / safe_num_tokens
        )
```

评论区精华

gemini-code-assist[bot] 提出了性能优化建议：num_tokens > 0 的 Python 三元表达式会触发 host-device 同步，建议改用 torch.clamp。作者未回复，但该建议合理。

- 性能：host-device 同步 (performance): 未采纳，但建议合理。PR 已合并，作者未修改。

风险与影响

- 风险：风险较低。变更仅影响 MTP 损失的梯度路径，且均增加了安全 clamp，不会改变正常训练行为。PR 已被合并者关闭（'fixed in megatron-bridge'），说明此修复在另一模块中已处理。
- 影响：影响范围小，仅修改 verl/models/mcore/mtp_patch.py 一个文件，影响使用 MTP 和上下文并行的 Megatron 训练场景。修复可防止偶发 NaN 和梯度错误，提升训练稳定性。
- 风险标记：代码已被上游合并替代

关联脉络

- PR #5782 [megatron, ckpt] fix: handle None param_data in get_megatron_module_device when use_distributed_optimizer=False: 同为 Megatron 模块的 bugfix，涉及分布式训练边界情况。
- PR #6562 [vllm, megatron] fix: mxfp8 trainning support on Ascend NPU: 同为 Megatron 模块的 bugfix，涉及训练稳定性的修复。