PR #6001 完整报告

verl-project/verl

[data] fix: pad data in preprocess_packed_seqs if shorter than align_size

合并时间：2026-04-16 16:04

原文链接：http://prhub.com.cn/verl-project/verl/pull/6001

执行摘要

- 一句话：修复序列预处理中数据长度不足导致的索引越界问题，确保上下文并行切片安全。
- 推荐动作：该 PR 值得精读，因为它展示了在分布式训练中处理数据对齐和边界条件的典型模式。关注点包括：
  - 如何安全地处理可变长度序列的切片，避免索引越界。
  - 在 review 讨论中，权衡了填充方案与索引检查方案的优缺点，最终选择了更轻量级的修复方式。
  - 可作为处理类似数据预处理边界问题的参考案例。

功能与动机

PR body 指出，在 preprocess_packed_seqs 函数中，如果序列元素数量小于对齐大小（align_size），切片操作 d[half_seqlen * cp_rank : half_seqlen * (cp_rank + 1)] 会导致索引越界。该问题已在 preprocess_thd_engine 中修复，本次变更旨在同步修复此函数，确保上下文并行处理中的安全性。

实现拆解

1. 识别问题点：在 verl/models/mcore/util.py 的 preprocess_packed_seqs 函数中，当 cp_size > 1 时，对输入张量 d 进行切片以支持上下文并行。原代码直接使用 half_seqlen * cp_rank 和 half_seqlen * (cp_rank + 1) 作为切片索引，未考虑 d 的实际长度可能小于这些索引值。
2. 修复切片逻辑：将原直接切片替换为显式计算切片边界 first_start、first_end 和 remain_start、remain_end，并使用 min 和 max 函数确保索引不越界且长度非负。具体变更包括：
   - 计算第一段切片的起始和结束索引，并限制在 d.shape[0] 范围内。
   - 计算剩余切片的起始和结束索引，同样限制在 d.shape[0] 范围内。
   - 使用 max(first_end - first_start, 0) 和 max(remain_end - remain_start, 0) 确保切片长度非负，避免负长度导致的错误。
3. 保持函数签名不变：函数输入输出接口未改变，仅内部逻辑调整，不影响外部调用。
4. 无测试或配置配套改动：本次变更仅涉及核心逻辑修复，未添加或修改测试文件、配置文件或部署脚本。

关键文件：

- verl/models/mcore/util.py（模块 模型工具；类别 source；类型 core-logic；符号 preprocess_packed_seqs）：这是本次 PR 的唯一变更文件，包含了修复序列预处理中索引越界问题的核心逻辑。

关键符号：preprocess_packed_seqs

关键源码片段

verl/models/mcore/util.py

这是本次 PR 的唯一变更文件，包含了修复序列预处理中索引越界问题的核心逻辑。

```
def preprocess_packed_seqs(
    input_ids: torch.Tensor,
    attention_mask: torch.Tensor,
    pre_process: bool = True,
    cp_size: int = 1,
    cp_rank: int = 0,
    use_fp8_padding: bool = False,
    align_size: int = 8
):
    # ... 省略前部分代码 ...
    if pre_process:
        input_ids_rmpad = torch.zeros(shape, dtype=input_ids.dtype, device=input_ids.device)
        for i in range(batch_size):
            # Use Python int, so no GPU→CPU sync in the loop
            if cp_size <= 1:
                seqlen = seqlens_in_batch_cpu[i]
                start_idx = cu_seqlens_padded_cpu[i]
                input_ids_rmpad[start_idx : start_idx + seqlen] = input_ids[i, attention_mask[i]]
                continue

            seqlen_padded_i = seqlens_in_batch_padded_cpu[i]
            seqlen = seqlen_padded_i // cp_size
            half_seqlen = seqlen // 2
            start_idx = cu_seqlens_padded_cpu[i] // cp_size
            # split to 2 chunks
            d = input_ids[i, attention_mask[i]]
            # 修复点：计算第一段切片边界，并确保不越界
            first_start = half_seqlen * cp_rank
            first_end = min(half_seqlen * (cp_rank + 1), d.shape[0]) # 限制结束索引不超过张量长度
            first_len = max(first_end - first_start, 0) # 确保长度非负
            if first_len > 0:
                input_ids_rmpad[start_idx : start_idx + first_len] = d[first_start:first_end]

            # 修复点：计算剩余切片边界，同样进行越界保护
            remain_start = seqlen_padded_i - half_seqlen * (cp_rank + 1)
            remain_end = seqlen_padded_i - half_seqlen * cp_rank
            remain_end = min(remain_end, d.shape[0]) # 限制结束索引
```

```
            remain_len = max(remain_end - remain_start, 0) # 确保长度非负
            if remain_len > 0:
                input_ids_rmpad[start_idx + half_seqlen : start_idx + half_seqlen + remain_len] = d[
                    remain_start:remain_end
                ]
    # ... 省略后部分代码 ...
```

评论区精华

review 中 gemini-code-assist[bot] 指出两个关键问题：

1. 对齐不足风险：仅填充到 *align_size* 可能不足以支持 FP8 填充场景，因为 *seqlen_padded_i* 可能远大于 *align_size*，导致切片仍可能越界。建议填充到 *seqlen_padded_i* 以确保安全。
2. 实现一致性：当前实现使用手动索引钳位和条件检查，与 PR 描述中“填充到对齐大小”的建议不一致，也不同于 *preprocess_thd_engine* 中的做法。建议采用填充方式以简化逻辑并保持代码库一致性。 最终提交的代码未采纳填充建议，而是通过索引边界检查和长度非负验证来修复问题，这可能基于性能或简化变更的考虑。

- 索引越界修复方案 (correctness): 最终代码未采纳填充建议，而是通过显式边界检查和长度非负验证来修复问题，可能基于实现复杂性或性能考虑。
- FP8 填充不足风险 (correctness): 此问题在代码中未直接处理，但通过索引边界检查缓解了越界风险，不过未根本解决 FP8 场景下的潜在问题。

风险与影响

- 风险：技术风险：
- 回归风险：修改了核心数据预处理路径，如果索引计算错误，可能导致数据错位或丢失，影响训练结果。但变更较小且逻辑清晰，风险较低。
- 性能影响：增加了 min 和 max 计算，可能引入微小开销，但在序列处理循环中影响可忽略。
- 兼容性：函数接口未变，与调用方兼容。
- 未解决疑虑：review 中提到的 FP8 填充不足问题未在代码中体现，如果未来启用 FP8 填充且序列长度不足，可能仍存在越界风险。
- 影响：影响范围：
- 用户影响：修复了潜在的数据处理崩溃问题，提升系统稳定性，用户在使用上下文并行训练时更可靠。
- 系统影响：仅影响 Megatron 模型中的序列预处理逻辑，涉及 verl/models/mcore/util.py 文件，不影响其他模块。
- 团队影响：为数据预处理路径提供了更健壮的边界处理，减少调试时间。
- 风险标记：核心路径变更，边界条件处理

关联脉络

- 暂无明显关联 PR