

PR #7702 完整报告

verl-project/verl

[megatron] fix: pad VLM THD input to TP/CP alignment before SP scatter

合并时间: 2026-09-11 10:17

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7702>

执行摘要

- 一句话: 对齐 VLM THD 序列长度, 修复 Megatron SP 崩溃
- 推荐动作: 建议精读, 尤其是正在维护 Megatron VLM / 序列并行 / 上下文并行前向路径的工程师。值得关注的三个设计点: 一是「SP scatter 要求 S 能被 tp_size 整除」这一硬约束通过 padding 在模型入口处满足, 而不是在 scatter 处打补丁; 二是 CP>1 时对齐粒度取 $tp_size * cp_size * 2$ 的推导 (zigzag 切分保证单 rank 局部长度仍可被 tp_size 整除); 三是把对齐逻辑收敛到 build_vlm_attn_mask_thd、让 fused 路径复用, 避免两套实现继续分叉。同时应把 Copilot 的测试建议当作待办跟进: 补一个覆盖 B=1、4026 这类奇数长度的 CPU 单测, 断言输出 S 可被 tp_size (CP>1 时为 $tp_size * cp_size * 2$) 整除, 并在 CP 分支校验每个 rank 拿到完整切片。

功能与动机

PR body 指出这是一个真实的训练崩溃: 当 `use_fused_kernels=True`、`use_dynamic_bsz=True`、`sequence_parallel=True` 同时开启时, 视觉分支用 `to_padded_tensor` 把嵌套 `input_ids` 转成 BSHD「without TP/CP alignment padding」, Bridge 随后把 `combined_embeddings` 展平成 [S, B, H] 调用 `scatter_to_sequence_parallel_region`, 而该算子断言 $S \% tp_size == 0$; 在 4026 tokens、TP=2 的场景下断言直接触发、任务崩溃。作者进一步定位到 B=1 (micro-batch size 1) 时 Bridge 的 `legacy_packed_bshd` 判定 (`input_ids.size(0) > 1`) 为假, 其内部本可做逐序列 TP 对齐的 `preprocess_packed_seqs` 被跳过, 未对齐的 `combined_embeddings` 一路走到 SP scatter。非 fused 路径经 `build_vlm_attn_mask_thd` 使用裸 `to_padded_tensor`, 存在同一 bug, 因此需要一处修复覆盖两条路径。

实现拆解

1. 根因定位 (PR body + 讨论): 崩溃发生在 SP scatter 的 $S \% tp_size == 0$ 断言上, 触发条件是 THD 打包序列的最大长度未按 TP/CP 对齐, 且 B=1 时 Megatron Bridge 跳过内部逐序列对齐逻辑。修复思路不是绕开 scatter, 而是让 padding 长度在进入模型前就满足对齐口径。
2. 对齐逻辑下沉到 `build_vlm_attn_mask_thd` (`verl/models/mcore/util.py`): 新增 `tp_size = mpu.get_tensor_model_parallel_world_size() / cp_size = mpu.get_context_parallel_world_size()` 读取, 按 $align_size = tp_size * cp_size * 2$ if $cp_size > 1$ else `tp_size` 计算对齐粒度, 对 `seqLens_in_batch.max()` 向上取整后显式传给

to_padded_tensor(pad_token_id, output_size=(batch_size, max_seqlen))。该口径与既有 build_vlm_attn_mask_bshd 完全一致 (CP>1 时因 zigzag 切分需要先满足 2*cp_size, 再满足 tp_size)。影响: attention_mask 随 padded 形状一起变大, 多出来的列全为 False, 不参与注意力。

3. fused 路径复用共享 helper (verl/models/mcore/model_forward_fused.py) : fused_forward_model_engine_inner 的 vision_model 分支删除内联的 input_ids.to_padded_tensor(pad_token_id) 与 torch.arange(max_seq_len) < seqLens_in_batch 手工掩码构造, 改为一处 input_ids_rmpad, attention_mask = build_vlm_attn_mask_thd(input_ids, pad_token_id), 并新增对应 import。收益有两点: fused 与非 fused 路径共享同一份对齐修复; input_ids_rmpad 与 attention_mask 由同一个 helper 产出, 形状天然一致, 避免两处独立构造导致的错位。
4. CP 分支 padding 下界修正 (preprocess_thd_engine, verl/models/mcore/util.py) : CP>1 的切分分支把 if d.shape[0] < align_size 改为 pad_target = max(align_size, seqLen_padded_i) 后再比较与填充。原因是只补到 align_size 可能仍短于该序列的 padded 长度 seqLen_padded_i, 会让 remain_end = min(remain_end, d.shape[0]) 提前截断, 导致部分 CP rank 拿不到完整切片。注意 logger.warning_once 的文案仍只打印 align_size, 与新的 pad_target 语义不完全对应 (Copilot 的 suppressed comment 也指出了这一点)。
5. 测试、配置与部署配套: 缺失。本 PR 未新增或修改任何测试文件, 也没有配置键、CLI 参数或 schema 变更。作者给出的验证是端到端证据: Qwen3.6-35B-A3B VLM Geo3K Megatron LoRA、4 节点 32x H20、merge=False, 修复前在 TP=2 + SP 下崩溃, 修复后 step1 pearson 0.9995、entropy 0.22、reward 0.65、390/390 adapter weights 流式传输成功。

关键文件:

- verl/models/mcore/util.py (模块 模型前向; 类别 source; 类型 data-contract; 符号 build_vlm_attn_mask_thd, preprocess_thd_engine, build_vlm_attn_mask_bshd) : 核心修复点: build_vlm_attn_mask_thd 增加 TP/CP 对齐 padding (与非 fused 的 build_vlm_attn_mask_bshd 口径一致), 并修正 preprocess_thd_engine 中 CP 切分的 padding 下界, 是本次崩溃的根因所在。
- verl/models/mcore/model_forward_fused.py (模块 模型前向; 类别 source; 类型 core-logic; 符号 fused_forward_model_engine_inner, build_vlm_attn_mask_thd) : fused 前向入口 fused_forward_model_engine_inner 的 vision 分支改用共享 helper, 删除内联的未对齐 to_padded_tensor 和手工 mask, 是让 fused 路径真正吃到对齐修复的关键一步。

关键符号: build_vlm_attn_mask_thd, preprocess_thd_engine, fused_forward_model_engine_inner, build_vlm_attn_mask_bshd

关键源码片段

verl/models/mcore/util.py

核心修复点: `build_vlm_attn_mask_thd` 增加 TP/CP 对齐 padding (与非 fused 的 `build_vlm_attn_mask_bshd` 口径一致), 并修正 `preprocess_thd_engine` 中 CP 切分的 padding 下界, 是本次崩溃的根因所在。

1) `build_vlm_attn_mask_thd`: 把 TP/CP 对齐补齐到 THD 路径

```
def build_vlm_attn_mask_thd(input_ids: torch.Tensor, pad_token_id: int = None):
    # input_ids 是 THD 布局的 jagged 嵌套张量, offsets() 的差分即每条序列的真实长度
    seqLens_in_batch = input_ids.offsets().diff()

    # 对齐到 TP/CP: combined_embeddings 在 SP scatter 前会被展平成 [S, B, H],
    # 只有 S % tp_size == 0 才不会触发 scatter_to_sequence_parallel_region 的断言;
    # 对齐口径与 build_vlm_attn_mask_bshd 保持一致。
    tp_size = mpu.get_tensor_model_parallel_world_size()
    cp_size = mpu.get_context_parallel_world_size()
    # CP 走 zigzag 切分时, 每条序列先按 2 * cp_size 对齐, 切分后单个 rank 的局部长度
    # 仍需被 tp_size 整除, 因此总长度必须是 tp_size * cp_size * 2 的倍数。
    align_size = tp_size * cp_size * 2 if cp_size > 1 else tp_size
    max_seqLen = int(seqLens_in_batch.max().item())
    if align_size > 1:
        max_seqLen += (align_size - max_seqLen % align_size) % align_size

    batch_size = input_ids.shape[0]
    # 显式传入 output_size, 避免沿用未对齐的原始最大长度 (旧实现就是这么崩的)
    input_ids_with_pad = input_ids.to_padded_tensor(pad_token_id, output_size=(batch_size, max_seqLen))
    attention_mask = torch.zeros_like(input_ids_with_pad, dtype=torch.bool)
    for i, seqLen in enumerate(seqLens_in_batch):
        # 只有真实 token 为 True, 对齐补出来的尾部位置不参与注意力
        attention_mask[i, :seqLen] = True

    return input_ids_with_pad, attention_mask
```

2) `preprocess_thd_engine` 的 CP 分支: 填充下界改为 `max(align_size, seqLen_padded_i)`

```
# CP 分支: 每条序列按 cp_size 切成 2 段, 每个 rank 取一整段
seqLen_padded_i = seqLens_in_batch_padded_cpu[i]
seqLen_orig_i = seqLens_in_batch_cpu[i]
seqLen = seqLen_padded_i // cp_size
half_seqLen = seqLen // 2
start_idx = cu_seqLens_padded_cpu[i] // cp_size
d = input_ids[i]

# 只补到 align_size 可能仍短于本序列的 padded 长度 seqLen_padded_i,
# 会让下方 remain 区间被 d.shape[0] 截断、部分 CP rank 拿不到完整片段,
# 因此取两者的最大值作为填充目标。
pad_target = max(align_size, seqLen_padded_i)
if d.shape[0] < pad_target:
    original_size = d.shape[0]
    pad = torch.zeros((pad_target - d.shape[0], *d.shape[1:]), dtype=d.dtype, device=d.
```

```

device)
d = torch.cat([d, pad], dim=0)
logger.warning_once(
    f"Padding tensor for context parallel alignment, original_size={original_size}, "
    f"align_size={align_size}" # 文案仍只提 align_size, 未反映 pad_target
)

# 第一段切片
input_ids_rmpad[start_idx : start_idx + half_seqlen] = d[
    half_seqlen * cp_rank : half_seqlen * (cp_rank + 1)
]

# 第二段 (remain) 切片, remain_end 不再被过短的 d 截断
remain_start = seqlen_padded_i - half_seqlen * (cp_rank + 1)
remain_end = min(seqlen_padded_i - half_seqlen * cp_rank, d.shape[0])
remain_len = remain_end - remain_start

```

verl/models/mcore/model_forward_fused.py

fused 前向入口 `fused_forward_model_engine_inner` 的 vision 分支改用共享 helper, 删除内联的未对齐 `to_padded_tensor` 和手工 mask, 是让 fused 路径真正吃到对齐修复的关键一步。

`fused_forward_model_engine_inner` 的 vision 分支: 统一走对齐 helper

```

attention_mask = None
if vision_model:
    # 视觉分支统一走 build_vlm_attn_mask_thd:
    # - 旧实现在这里直接 input_ids.to_padded_tensor(pad_token_id), 再用
    #   arange(max_seq_len) < seqLens_in_batch 手工造 mask, 最大长度未对齐;
    # B=1 时 Megatron Bridge 的 legacy_packed_bshd 判定 input_ids.size(0) > 1
    # 为假, 内部逐序列 TP 对齐的 preprocess_packed_seqs 被跳过,
    # combined_embeddings 会以未对齐的 S 走到 SP scatter 触发断言崩溃;
    # - 现在 input_ids_rmpad 与 attention_mask 由同一个 helper 产出, 形状天然一致,
    # 并与非 fused 路径共享同一份 TP/CP 对齐口径。
    input_ids_rmpad, attention_mask = build_vlm_attn_mask_thd(input_ids, pad_token_id)

labels_rmpad, _, _ = preprocess_thd_engine(
    labels,
    pre_process=True,
    need_roll=True,
    use_fp8_padding=use_fp8_padding,
    min_local_rows=min_local_rows,
    pad_to_length_bucket=pad_to_length_bucket,
    cp_layout=cp_layout,
    local_cp_size=local_cp_size,
)
labels_rmpad = labels_rmpad.contiguous()
forward_kwargs = dict(
    input_ids=input_ids_rmpad,
    attention_mask=attention_mask,

```

```
position_ids=None,
packed_seq_params=packed_seq_params,
labels=labels_rmpad,
**model_kwargs,
)
```

评论区精华

本轮 review 讨论量很小：1 条 Copilot review comment + 1 次人工 approve。最有价值的两点是：

1. 测试覆盖缺口 (Copilot, [verl/models/mcore/util.py:862](#))：认可「对齐 padding 修复了真实的崩溃模式」，但强调「there is no unit test asserting the padded `max_seqlen` is divisible by `tp_size` (or `tp_size * cp_size * 2` when `CP>1`)」，并建议构造 `B=1 + 4026` 这类奇数 token 数的用例。该意见未被解决，PR 在 wuxibin89 approve 后直接合入。
2. CP padding 下界与告警文案：Copilot 在 review body 的 suppressed comment 中指出 [verl/models/mcore/util.py:488](#) 处的 warning 「claims padding is only for `align_size`, but the new logic pads to...」，即日志文案与实际逻辑脱节，属可读性 / 可观测性瑕疵，未阻塞合入。

人工 reviewer wuxibin89 给出 APPROVED 且无文字评论，说明维护者对「把对齐逻辑收敛到共享 helper」这一方向没有异议。

- 缺少 TP/CP 对齐的单测覆盖 (testing): 未在该 PR 中补充测试，作者以 Qwen3.6-35B-A3B VLM Geo3K 的 4 节点 E2E 结果 (step1 pearson 0.9995) 作为验证证据；wuxibin89 approve 后 PR 合入，测试作为遗留待办。
- CP 分支 padding 下界从 `align_size` 改为 `max(align_size, seqlen_padded_i)` (correctness): 逻辑本身被接受并合入，仅日志文案的准确性问题被记为小瑕疵，未阻塞合并。
- fused 路径改用共享 helper 的设计取舍 (design): 维护者 wuxibin89 直接 APPROVED 且未提出异议，方向被认可。

风险与影响

- 风险：
 1. 缺少回归测试 (最高优先级)：`build_vlm_attn_mask_thd` 现在同时服务 fused 与非 fused 两条 VLM 前向路径，却没有单测锁定 `max_seqlen % tp_size == 0` (`CP>1` 时为 `tp_size * cp_size * 2`) 这一不变量。Copilot 已经点名这个风险，PR 仍合入，后续任何改动都可能静默破坏对齐，且只有在特定 `B=1 + 奇数 token` 数场景下才暴露为崩溃。
 1. 共享 helper 扩大了影响面：`model_forward_fused.py` 从内联构造改为调用 `build_vlm_attn_mask_thd` 后，fused 路径的 mask 构造方式发生变化——旧的 `torch.arange(...) < seqLens_in_batch.unsqueeze(1)` 是设备端向量化（且显式 `to(input_ids.device)`），新实现是 `torch.zeros_like(..., dtype=torch.bool)` 加 CPU 侧 Python 循环逐行填充。功能等价且开销可忽略，但属于同一 PR 里的隐式行为变更，若 mask dtype/ 设备语义有细微差异，影响的是整个 VLM fused 前向。

2. padded 长度变化带来的下游形状约束: `input_ids_with_pad` 的 S 变大后, 需保证 `postprocess_thd_engine` 侧按 `seq_lens_in_batch` 切片 / 还原的逻辑仍能对齐; `build_vlm_attn_mask_bshd` 中专门有 `forced_max_seq_len` 参数来保证 `logits.shape[:2] == label.shape[:2]`, 说明这类 padded 维度在 BSHD 路径上确实需要与 label 严格对齐, 而 THD 路径本次没有对应的显式约束或断言。
3. CP 分支填充语义变化: `pad_target = max(align_size, seq_len_padded_i)` 会让更多序列被补零到 `seq_len_padded_i`, 补出来的内容本身是 padding (`position_ids` 已按 `seq_len_orig_i` 截断), 因此不改变有效 token, 但确实改变了进入 CP 切分的张量长度, 属于该分支的隐性契约调整, 同样无单测覆盖。
4. 性能影响可忽略: 每个 batch 最多多算 `align_size - 1` ($CP > 1$ 时为 `tp_size * cp_size * 2 - 1`) 个 token 的 embedding 与注意力掩码开销, 相对 4 节点规模的训练可以忽略。
 - 影响: 用户侧: 修复合乎条件的 VLM Megatron 训练任务从「必崩」变为「可跑」, 影响面是一组明确受限的配置组合——VLM (`vision_model=True`) + `use_fused_kernels=True` + `use_dynamic_bsz=True` + `sequence_parallel=True` (或 $CP > 1$), 典型场景是 Qwen3.6-35B-A3B VLM Geo3K Megatron LoRA 这类多模态 RL 训练; 不满足该组合的用户完全不受影响, 且无 API/ 配置破坏性变更。系统侧: 对齐口径在 THD 与 BSHD 之间完成统一, 减少了同一不变量在两处实现漂移的可能, 但也把两条前向路径耦合到同一个 helper 上, 出问题时爆炸半径同时覆盖 fused 与非 fused。团队侧: 这是一次「先 E2E 验证、后补测试」的合入, 维护者需要在后续补上针对 `build_vlm_attn_mask_thd` 的 CPU 单测 (仓库已有大量 `tests/utils/*_on_cpu.py` 风格用例), 否则该修复的长期稳定性依赖人工回归。
 - 风险标记: 缺少测试覆盖, 核心前向路径变更, 多路径共享 helper 影响面扩大, padded 维度与下游形状契约未加断言

关联脉络

- PR #7106 [megatron] fix: preserve R2 router replay for THD-packed batches: 同属 Megatron THD-packed batch 前向路径的修复, 涉及 `verl/utils/megatron_utils.py` 与 `verl/workers/engine/megatron/transformer_impl.py`, 与本次 THD 输入对齐改动处于同一段工程链路, 容易在 packed 序列处理上互相影响。
- PR #7786 Revert "[megatron] fix: preserve R2 router replay for THD-packed batches": PR#7106 的回滚, 说明 Megatron THD 打包前向路径近期仍处于不稳定 / 调整状态; 在这样活跃的路径上落一个无单测的 padding 语义变更, 回归风险更高, 值得与本次改动一并观察。