PR #7261 完整报告

verl-project/verl

[megatron] fix: pad multidimensional THD tensors along the sequence dimension

合并时间：2026-08-04 23:19

原文链接：http://prhub.com.cn/verl-project/verl/pull/7261

执行摘要

- 一句话：修复 THD 多维张量在 zigzag CP 下的序列维补零缺失
- 推荐动作：值得精读。这是一个小改动揭示典型一维假设问题的好案例：8 行修复解决了多维张量在 CP 下的系统性崩溃，且通过差分测试（2304 组对比）严格保证一维行为不变。值得关注的设计决策：reviewer 对作者分支方案的纠偏（更简单的统一逻辑通常更正确）、以及"用 shape[0] 而不是 numel 做序列长度判断"的普适性原则，对后续接触 THD/CP 预处理的工程师有直接参考价值。

功能与动机

PR body 明确描述了崩溃现象：RuntimeError: The expanded size of the tensor (1) must match the existing size (0) at non-singleton dimension 0. Target sizes: [1, 48, 8]. Tensor sizes: [0, 48, 8]。根因是原补零逻辑按一维输入设计（d.shape = [seq_len] 时 d.numel() == d.shape[0]），而同一函数也被 Router Replay 等多维 per-token 元数据使用（d.shape = [seq_len, num_layers, topk]），导致 CP 对齐与 chunk 切分沿 dim 0 进行时补零被跳过，最终 CP rank 读取空序列切片触发崩溃。修复目标是让多维张量也获得形状保持的序列维补零，同时不改变一维输入的既有契约。

实现拆解

1. 根因定位：在 verl/models/mcore/util.py 的 preprocess_thd_engine 的 zigzag CP 分支（cp_size > 1）中，补零判断 if d.numel() < align_size 隐含一维假设。对形状为 [1, 48, 8] 的张量，d.numel() = 384，align_size = 4，条件恒为假，补零被跳过；而后续 remain 段切分按 dim 0 进行，remain_end = min(remain_end, d.shape[0]) 会读到长度为 0 的切片并赋给形状 [1, 48, 8] 的目标，触发 RuntimeError。
2. 修复核心：将判定条件改为 d.shape[0] < align_size，补零张量形状改为 (align_size - d.shape[0], *d.shape[1:])，即沿序列维度补零并保留全部 trailing 维度；一维时 d.shape[0] == d.numel()，判定与补零结果数学等价，行为完全保持。warning 日志同步从 numel 改为 shape[0]。
3. review 演进：作者最初方案（PR body 所述）按维度区分补零目标 pad_target = align_size if d.dim() == 1 else seqlen_padded_i，reviewer wuxibin89 指出该逻辑不正确并直接给出统一按 align_size 补零的写法，最终合入版本采用 reviewer 方案，比草案更简洁且语义更一致。

4. 测试配套：tests/utils/test_megatron_bshd_preprocess.py 新增参数化测试覆盖多维 Router Replay 数据在 cp_size=2 下两个 rank 的补零与切分结果，以及一维 need_roll=True 时 zigzag label 滚动的行为保持。PR body 报告了 2304 组差分对比（序列长度 1-64、TP 1/2/4、CP 2/4、全部 CP rank、need_roll 开关），一维行为零差异；全量相关测试 48 个通过。

关键文件：

- verl/models/mcore/util.py（模块 CP 预处理；类别 source；类型 data-contract；符号 preprocess_thd_engine）：修复核心所在：将 CP 对齐补零从按标量总数（numel）判断改为按序列维度（shape[0]）判断，并让补零张量保留 trailing 维度，直接修复多维 per-token 元数据在 zigzag CP 下的 shape mismatch 崩溃。
- tests/utils/test_megatron_bshd_preprocess.py（模块 CP 预处理；类别 test；类型 test-coverage；符号 test_preprocess_thd_engine_pads_multidimensional_router_data, test_preprocess_thd_engine_preserves_1d_zigzag_roll_alignment）：新增两类回归测试：多维 Router Replay 数据按序列维补零后各 CP rank 均拿到形状正确的 chunk；一维 label 在 need_roll=True 下 zigzag 滚动行为保持，防止修复破坏旧契约。

关键符号：preprocess_thd_engine, test_preprocess_thd_engine_pads_multidimensional_router_data, test_preprocess_thd_engine_preserves_1d_zigzag_roll_alignment

关键源码片段

verl/models/mcore/util.py

修复核心所在：将 CP 对齐补零从按标量总数（numel）判断改为按序列维度（shape[0]）判断，并让补零张量保留 trailing 维度，直接修复多维 per-token 元数据在 zigzag CP 下的 shape mismatch 崩溃。

```
# verl/models/mcore/util.py 中 preprocess_thd_engine 的 zigzag CP 分支（cp_size > 1）
# 对当前 batch 内第 i 条序列做 CP 对齐与 chunk 切分
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
# split to 2 chunks
d = input_ids[i]

# 修复前：用 d.numel() < align_size 判断，对多维张量 [1, 48, 8] 来说
# numel = 384 >= 4 恒成立，补零被跳过，后续 remain 段读到空切片 [0, 48, 8]
# 修复后：按序列维度（dim 0）判断与补零，保留 trailing 维度（num_layers、topk）
if d.shape[0] < align_size:
    pad_shape = (align_size - d.shape[0], *d.shape[1:])
    pad = torch.zeros(pad_shape, 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={d.shape[0]}, "
        f"align_size={align_size}"
```

```
    )

# zigzag 切分：前半段由 cp_rank 决定，后半段反向读取（remain 段）
input_ids_rmpad[start_idx : start_idx + half_seqlen] = d[
    half_seqlen * cp_rank : half_seqlen * (cp_rank + 1)
]

# Build position_ids for the first chunk
position_ids_rmpad[start_idx : start_idx + half_seqlen] = torch.arange(
    half_seqlen * cp_rank, half_seqlen * (cp_rank + 1), dtype=torch.long, device=input_ids.device
)

remain_start = seqlen_padded_i - half_seqlen * (cp_rank + 1)
remain_end = seqlen_padded_i - half_seqlen * cp_rank
# 用 d.shape[0] 而非 numel 截断，配合修复后的补零保证多维张量也能安全切片
remain_end = min(remain_end, d.shape[0])
remain_len = remain_end - remain_start
if remain_len > 0:
    input_ids_rmpad[start_idx + half_seqlen : start_idx + half_seqlen + remain_len] = d[
        remain_start:remain_end
    ]
    # Build position_ids for the remaining chunk: use remain_start as base,
    # clamped to original seqlen to avoid exceeding seqlen-1 for padded positions
    pos_end = min(remain_end, seqlen_orig_i)
    valid_pos_len = pos_end - remain_start
    if valid_pos_len > 0:
        position_ids_rmpad[start_idx + half_seqlen : start_idx + half_seqlen + valid_pos_len] = (
            torch.arange(remain_start, pos_end, dtype=torch.long, device=input_ids.device)
        )
```

tests/utils/test_megatron_bshd_preprocess.py

新增两类回归测试：多维 Router Replay 数据按序列维补零后各 CP rank 均拿到形状正确的 chunk；一维 label 在 need_roll=True 下 zigzag 滚动行为保持，防止修复破坏旧契约。

```
# tests/utils/test_megatron_bshd_preprocess.py 新增回归测试

@pytest.mark.parametrize("cp_rank", [0, 1])
def test_preprocess_thd_engine_pads_multidimensional_router_data(monkeypatch, cp_rank):
    mcore_util = _load_mcore_util_with_stubbed_megatron(
        monkeypatch,
        tp_size=1,
        cp_size=2,
        cp_rank=cp_rank,
    )
    # Router Replay 的 per-token 元数据：1 个 token、48 层、top-8 专家
    route = torch.arange(48 * 8, dtype=torch.long).reshape(1, 48, 8)
    routed_experts = _nested_tensor([route])

    local_routes, packed_seq_params, _ = mcore_util.preprocess_thd_engine(routed_experts)
```

```
    # 序列长度 1 补零到 4（align_size），cp_size=2 时每个 rank 分到 2 行
    expected = torch.zeros((2, 48, 8), dtype=torch.long)
    if cp_rank == 0:
        expected[0] = route[0]
    assert local_routes.shape == (1, 2, 48, 8)
    assert packed_seq_params.cu_seqlens_q_padded.tolist() == [0, 4]
    torch.testing.assert_close(local_routes[0], expected)


@pytest.mark.parametrize(
    ("cp_rank", "expected"),
    [
        (0, [2, 3, 1, 1]),
        (1, [4, 5, 6, 7]),
    ],
)
def test_preprocess_thd_engine_preserves_1d_zigzag_roll_alignment(monkeypatch, cp_rank,
expected):
    # 回归保护：一维 label 在 need_roll=True 时的 zigzag 滚动边界行为必须保持
    mcore_util = _load_mcore_util_with_stubbed_megatron(
        monkeypatch,
        tp_size=1,
        cp_size=2,
        cp_rank=cp_rank,
    )
    labels = _nested_tensor([torch.arange(1, 8, dtype=torch.long)])

    local_labels, packed_seq_params, _ = mcore_util.preprocess_thd_engine(labels, need_roll=
    True)

    assert packed_seq_params.cu_seqlens_q_padded.tolist() == [0, 8]
    torch.testing.assert_close(local_labels[0], torch.tensor(expected, dtype=torch.long))
```

评论区精华

唯一的 review 评论发生在 verl/models/mcore/util.py：wuxibin89 直接否定了作者在 PR body 中描述的分支方案，指出 pad_target 不正确并给出修正代码。作者接受修改，最终合入代码与 PR body 描述不一致，以合入代码为准。这一纠偏说明：对于本场景，统一将补零目标对齐到 align_size 比按维度区分目标更简单且正确，作者草案中的 seqlen_padded_i 分支属于过度设计。

- pad_target 分支方案被 reviewer 纠正 (design): 合入版本采用 reviewer 建议，统一按序列维度与 align_size 对齐补零；PR body 中描述的分支逻辑与实际合入代码不一致，以合入代码为准。

风险与影响

- 风险：

1. 核心路径变更：preprocess_thd_engine 是 Megatron 后端 THD 输入预处理的核心函数，zigzag CP 布局下每个序列都会经过该分支，改动影响面广。
2. 数据契约变更：多维张量（Router Replay route、teacher top-k 等）从“不补零”变为“沿序列维补零”，依赖方需确认 shape 语义；但对一维输入数学等价，d.shape[0] == d.numel()，且测试覆盖一维差分对比 2304 组零差异。
3. 日志语义变化：warning 日志 original_size 从 numel 变为 shape[0]，一维场景数值不变，多维场景数值变小，仅影响日志可读性。
4. 超短序列边界：修复只保证 d.shape[0] >= align_size，对 align_size <= shape[0] < seqlen_padded_i 的中间区间仍依赖 remain_end = min(remain_end, d.shape[0]) 截断保护，行为与一维路径一致。 - 影响：影响范围集中在使用 Megatron 后端 + CP（尤其 zigzag 布局）+ 多维 per-token 元数据（Router Replay、蒸馏 teacher top-k）的训练作业：此前这类组合在短序列下必然崩溃，修复后各 CP rank 能获得形状正确的 chunk。输入 IDs、labels、loss masks、replay masks 等一维数据路径行为不变；cp_size <= 1、contiguous CP 布局不受影响。无公共 API 或配置变更，对用户是透明的 bugfix。对团队的启示是：同一预处理函数被一维 token 数据和多维元数据共用，任何长度 / 对齐假设都需要按维度显式区分。 - 风险标记：核心路径变更 , 数据契约变更

关联脉络

- PR #5410 引入原始一维 padding 逻辑：PR body 明确说明本 PR 修复的原始一维补零逻辑由 #5410 引入，是本次 bug 的源头。
- PR #6703 一维 FP8 + CP padding 相关修复：PR body 提到 #6703 处理的是同区域一维 FP8 + CP 补零问题，与本 PR 共用 preprocess_thd_engine 的补零路径。
- PR #7221 [megatron] feat: support contiguous context-parallel layout for DeepSeek V4: 同期改动 verl/models/mcore/util.py 与 tests/utils/test_megatron_bshd_preprocess.py，为 DeepSeek V4 引入 contiguous CP 布局，与本 PR 共同完善 Megatron 后端 CP 布局支持。
- PR #7242 [veomni] feat: add DeepSeek V4 support: 涉及 Router Replay 数据链路及同一批测试文件（test_router_replay_engine_helpers_on_cpu.py），本 PR 修复的多维 Router Replay 张量正是该特性的配套数据。