PR #6506 完整报告

verl-project/verl

[megatron, trainer] fix: preserve BSHD top-k distillation shape

合并时间：2026-05-28 10:57

原文链接：http://prhub.com.cn/verl-project/verl/pull/6506

执行摘要

- 一句话：保留 teacher top-k 张量 dense 维度，修复 Megatron OPD 蒸馏预处理崩溃
- 推荐动作：值得精读的 bugfix PR，展示了在 nested tensor 预处理中如何合理保留末尾维度。核心设计模式可复用。重点关注 preprocess_bshd_engine 的 docstring 更新和 dense_shape 的用法。

功能与动机

来自 Issue #6492：OPD losses error for Megatron，teacher_topk_log_probs 形状 [b, length, 64] 但 preprocess_bshd_engine 函数预期 [n, length]。需要修改预处理函数以支持 teacher topk 张量的 dense trailing dimensions。

实现拆解

1. 在 verl/models/mcore/util.py 的 preprocess_bshd_engine 中提取 dense_shape = input_ids.shape[2:]，使得后续所有 padded tensors 分配时考虑额外维度（如 topk）。
2. 修改 input_ids_bshd 和 seq_padded 的创建，从形状 (batch, seq) 改为 (batch, seq, *dense_shape)，以容纳 topk 等信息。
3. 修改 position_ids 的扩展目标：从 expand_as(input_ids_bshd) 改为 expand_as(attention_mask)，避免因 input_ids_bshd 尺寸变化导致的维度不匹配。
4. 新增 tests/utils/test_megatron_bshd_preprocess.py，通过 monkeypatch 模拟 megatron 环境，提供 CPU/GPU 回归测试验证 1D 和 topk 输入的正确预处理。
5. 扩展 tests/utils/test_special_megatron_kl_loss_tp.py，添加 pad_for_bshd_preprocess 辅助函数和 BSHD 格式的 VP KL 正确性验证，确保与 THD 参考路径一致。

关键文件：

- verl/models/mcore/util.py（模块 BSHD 预处理；类别 source；类型 data-contract；符号 preprocess_bshd_engine）：核心代码变更，修复 preprocess_bshd_engine 以保留 dense trailing dimensions，是 bug 的直接修复点。
- tests/utils/test_megatron_bshd_preprocess.py（模块 预处理测试；类别 test；类型 test-coverage；符号 _load_mcore_util_with_stubbed_megatron, _nested_tensor, _check_topk_preprocess, test_preprocess_bshd_engine_preserves_1d_input_shape_on_cpu）：新增测试文件，提供 CPU/GPU 回归测试，覆盖 1D 和 topk 输入，确保预处理函数正确性。

- tests/utils/test_special_megatron_kl_loss_tp.py（模块 TP 蒸馏测试；类别 test；类型 test-coverage；符号 pad_for_bshd_preprocess）：扩展现有 TP KL 测试，添加 BSHD 格式的正确性验证，确保 loss 计算与 THD 参考一致。

关键符号：preprocess_bshd_engine, pad_for_bshd_preprocess, _load_mcore_util_with_stubbed_megatron, _check_topk_preprocess, test_preprocess_bshd_engine_preserves_1d_input_shape_on_cpu, test_preprocess_bshd_engine_preserves_topk_dense_dim_on_cpu, test_preprocess_bshd_engine_preserves_topk_dense_dim_on_gpu

关键源码片段

verl/models/mcore/util.py

核心代码变更，修复 preprocess_bshd_engine 以保留 dense trailing dimensions，是 bug 的直接修复点。

```
def preprocess_bshd_engine(
    input_ids: torch.Tensor, pre_process: bool = True, need_roll: bool = False, use_fp8_padding:
    bool = False
):
    """
    Preprocess bshd sequences.
    Returns (input_ids, attention_mask, position_ids).

    The input is a jagged nested tensor with shape [batch, seq, ...].
    Any dense dimensions after seq are preserved in the returned padded tensor.
    """
    cp_size = mpu.get_context_parallel_world_size()
    cp_rank = mpu.get_context_parallel_rank()

    batch_size = input_ids.shape[0]
    # 关键：保留 trailing dims，例如 topk=64 时 shape[2:] = (64,)
    dense_shape = tuple(input_ids.shape[2:])
    seqlens_in_batch = input_ids.offsets().diff()
    max_seqlen = seqlens_in_batch.max().item()
    tp_size = mpu.get_tensor_model_parallel_world_size()
    # ... (alignment logic unchanged, omitted for brevity) ...
    align_size = tp_size * cp_size * 2 if cp_size > 1 else tp_size
    if align_size > 1:
        pad_size = (align_size - max_seqlen % align_size) % align_size
        max_seqlen += pad_size

    local_max_seqlen = max_seqlen // cp_size if cp_size > 1 else max_seqlen
    attention_mask = torch.zeros(batch_size, local_max_seqlen, dtype=torch.bool, device=input_
    ids.device)
    # 原为 (batch_size, local_max_seqlen)，现在加入 *dense_shape 以容纳 topk 等维度
    input_ids_bshd = torch.zeros(
        (batch_size, local_max_seqlen, *dense_shape), dtype=input_ids.dtype, device=input_ids.
```

```
    device
)
seqlens_in_batch_cpu = seqlens_in_batch.tolist()
for i in range(batch_size):
    seqlen_i = int(seqlens_in_batch_cpu[i])
    if cp_size <= 1:
        attention_mask[i, :seqlen_i] = True
        input_ids_bshd[i, :seqlen_i] = input_ids[i] # input_ids[i] 形状为 [seqlen_i, *dense_shape]
        continue
    # CP zigzag handling (dense_shape preserved)
    seq = input_ids[i]
    if seqlen_i < max_seqlen:
        seq_padded = torch.zeros((max_seqlen, *dense_shape), dtype=seq.dtype, device=seq.
        device)
        seq_padded[:seqlen_i] = seq
        seq = seq_padded
    chunk_len = max_seqlen // (2 * cp_size)
    first_start = cp_rank * chunk_len
    second_start = (2 * cp_size - cp_rank - 1) * chunk_len
    first_chunk = seq[first_start : first_start + chunk_len]
    second_chunk = seq[second_start : second_start + chunk_len]
    local_seq = torch.cat((first_chunk, second_chunk), dim=0)
    input_ids_bshd[i] = local_seq
    # attention_mask remains 2D, unchanged
    valid_first = max(0, min(seqlen_i - first_start, chunk_len))
    valid_second = max(0, min(seqlen_i - second_start, chunk_len))
    if valid_first > 0:
        attention_mask[i, :valid_first] = True
    if valid_second > 0:
        attention_mask[i, chunk_len : chunk_len + valid_second] = True

# position_ids 现在扩展为 attention_mask 的形状而非 input_ids_bshd，保持 2D
if cp_size <= 1:
    position_ids = torch.arange(local_max_seqlen, dtype=torch.long, device=input_ids.device)
    position_ids = position_ids.unsqueeze(0).expand_as(attention_mask)
else:
    chunk_len = max_seqlen // (2 * cp_size)
    first_pos = torch.arange(first_start, first_start + chunk_len, dtype=torch.long, device=
    input_ids.device)
    second_pos = torch.arange(second_start, second_start + chunk_len, dtype=torch.long,
    device=input_ids.device)
    position_ids = torch.cat((first_pos, second_pos), dim=0).unsqueeze(0).expand_as(attention_
    mask)

if need_roll and cp_size <= 1:
    input_ids_bshd = torch.roll(input_ids_bshd, shifts=-1, dims=1)

return input_ids_bshd, attention_mask, position_ids
```

tests/utils/test_megatron_bshd_preprocess.py

新增测试文件，提供 CPU/GPU 回归测试，覆盖 1D 和 topk 输入，确保预处理函数正确性。

```
def _check_topk_preprocess(monkeypatch, device: torch.device):
    """Helper: verify that preprocess_bshd_engine preserves topk dense dimension."""
    mcore_util = _load_mcore_util_with_stubbed_megatron(monkeypatch)
    topk = 64

    # 构造两个样本，每个样本是 (seq_len, topk) 的张量
    logprob_rows = [
        torch.arange(3 * topk, dtype=torch.float32, device=device).reshape(3, topk),
        torch.arange(2 * topk, dtype=torch.float32, device=device).reshape(2, topk) + 1000,
    ]
    teacher_logprobs = torch.nested.as_nested_tensor(logprob_rows, layout=torch.jagged)

    logprobs_bshd, attention_mask, position_ids = mcore_util.preprocess_bshd_engine(teacher_
    logprobs)

    # 验证 shape 为 (batch, padded_seq_len, topk)
    assert logprobs_bshd.shape == (2, 4, topk)
    assert logprobs_bshd.device.type == device.type
    assert attention_mask.device.type == device.type
    assert position_ids.shape == (2, 4) # attention mask 和 position ids 仍是 2D

    # 验证实际值：有效 token 部分一致，padding 部分为零
    torch.testing.assert_close(logprobs_bshd[0, :3], logprob_rows[0])
    torch.testing.assert_close(logprobs_bshd[1, :2], logprob_rows[1])
    torch.testing.assert_close(logprobs_bshd[0, 3], torch.zeros(topk, dtype=torch.float32, device=
    device))
    torch.testing.assert_close(logprobs_bshd[1, 2:], torch.zeros(2, topk, dtype=torch.float32,
    device=device))

    # 同样验证 id 输入（int64）也走相同路径
    id_rows = [
        torch.arange(3 * topk, dtype=torch.long, device=device).reshape(3, topk),
        torch.arange(2 * topk, dtype=torch.long, device=device).reshape(2, topk) + 2000,
    ]
    teacher_ids = torch.nested.as_nested_tensor(id_rows, layout=torch.jagged)
    ids_bshd, ids_attention_mask, _ = mcore_util.preprocess_bshd_engine(teacher_ids)
    assert ids_bshd.shape == (2, 4, topk)
    assert ids_bshd.dtype == torch.long


def test_preprocess_bshd_engine_preserves_topk_dense_dim_on_cpu(monkeypatch):
    _check_topk_preprocess(monkeypatch, torch.device("cpu"))
```

tests/utils/test_special_megatron_kl_loss_tp.py

扩展现有 TP KL 测试，添加 BSHD 格式的正确性验证，确保 loss 计算与 THD 参考一致。

```
def pad_for_bshd_preprocess(self, tensor: torch.Tensor) -> torch.Tensor:
    """Mirror preprocess_bshd_engine's CP=1 sequence padding for student logits."""
    assert mpu.get_context_parallel_world_size() == 1, "This TP KL test does not initialize context
    parallelism"
    align_size = mpu.get_tensor_model_parallel_world_size()
    pad_size = (align_size - tensor.shape[1] % align_size) % align_size
    if pad_size == 0:
        return tensor

    # 保留 tensor 的所有尾部维度（如 vocab_size），只对序列维度补零
    padded = torch.zeros(
        tensor.shape[0],
        tensor.shape[1] + pad_size,
        *tensor.shape[2:],
        dtype=tensor.dtype,
        device=tensor.device,
    )
    padded[:, : tensor.shape[1]] = tensor
    return padded

# 在 verify_correctness 中增加的 BSHD 路径验证（节选）
full_student_logits_bshd_padded = self.pad_for_bshd_preprocess(full_student_logits_bshd)
vp_logits_bshd = full_student_logits_bshd_padded[..., shard_start:shard_end].contiguous().
detach().requires_grad_(True)
loss_out_bshd = compute_forward_kl_topk_vp(
    student_logits=vp_logits_bshd,
    teacher_topk_log_probs=teacher_topk_logps,
    teacher_topk_ids=teacher_topk_ids,
    config=cfg,
    data_format="bshd",
)
# 比较 BSHD 与 THD 的 loss 和梯度
```

评论区精华

Review 无实质性讨论。PR 作者通过 Issue #6492 清晰描述了 bug，设计决策在 PR body 中说明。wuxibin89 直接批准，变更被快速接受。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险：1）CP>1 路径的 position_ids 变更仅在 CP=1 测试下覆盖，但逻辑相似，风险低。2）topk 维度保留后，下游 loss 计算已验证一致性。3）完全向后兼容 1D BSHD 输入。4）无配置或 API 变更，不暴露用户。
- 影响：影响范围限于使用 Megatron 引擎且开启 topk 蒸馏（OPD）的用户。修复后 teacher topk 数据可正常预处理，之前会抛出形状异常。对其他模块无影响。团队需确保 CI 中扩展的测试通过。

- 风险标记：核心路径变更，CP 路径未覆盖，position_ids 扩展变更

关联脉络

- PR #6469 add top-k distillation overlap metrics: 同一测试文件 test_special_megatron_kl_loss_tp.py 被扩展以覆盖 BSHD 路径，且都涉及 topk 蒸馏的正确性验证。