PR #33322 完整报告

vllm-project/vllm

[Bugfix] Fix SP pass for multimodal models and PP+SP residual handling

合并时间：2026-05-10 10:44

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/33322

执行摘要

- 一句话：修复多模态模型 SP 和 PP+SP residual 处理 bug
- 推荐动作：值得精读的设计：residual 切片使用 tp_rank 感知的索引，以及 sync_and_gather_intermediate_tensors 中通过 all-gather 保证 SP + PP 兼容性。此外，团队对三种方案的权衡分析展现了良好的设计思维。建议关注后续 #36823 和 MoE SP 相关 PR。

功能与动机

部署多模态模型或带有 --enable-prompt-embeds 的文本模型时，Sequence Parallelism 因 torch compile 错误无法启用。具体错误包括：

1) aten.copy_ 形状不匹配（sharded vs full length）； 2) 所有 TP rank 使用相同的 residual 切片（residual[0:local_len]），导致结果错误。

实现拆解

1. 修复 residual 切片逻辑：在 sequence_parallelism.py 的 MiddleAllReduceRMSNormPattern.replacement 中，将 residual[0:reduce_scatter.size(0), ...] 改为 residual[self.tp_rank * local_len : (self.tp_rank + 1) * local_len, ...]，实现 TP rank 感知切片。
2. 重构 gpu_model_runner 中的 intermediate tensors 处理方法：将 sync_and_slice_intermediate_tensors 重写为 sync_and_gather_intermediate_tensors，在 SP 模式下对 residual 执行 all-gather 而非简单切片，确保下游 QKV + Attention 能访问完整 residual。
3. 支持 PP+SP 共存：在 PP 阶段间通过 sync_and_gather_intermediate_tensors 传递 intermediate tensors，通过 all-gather 正确恢复每个 rank 需要的残差数据。
4. 移除过渡性 InputMutationSlicingPass：借助 #36823 将 fused_add_rms_norm 纳入 vllm IR，从根源避免 AOT Autograd 插入形状冲突的 copy_，简化实现。
5. 测试配套：新增 E2E 测试 test_tp_sp_generation_prompt_embeds 覆盖 prompt_embeds 路径，包括 SP only（pp=1）和 SP+PP（pp=2）场景；在 _compare_sp 中添加 enable_prompt_embeds 参数。

关键文件：

- vllm/v1/worker/gpu_model_runner.py（模块 GPU 模型运行器；类别 source；类型 data-contract；符号 sync_and_slice_intermediate_tensors, sync_and_gather_intermediate_tensors）：核心变更文件：将 sync_and_slice_intermediate_tensors 重构为 sync_and_gather_intermediate_tensors，在 SP 模式下对 residual 进行 all-gather 而非简单切片，确保完整语义。
- vllm/compilation/passes/fusion/sequence_parallelism.py（模块 编译优化器；类别 source；类型 dependency-wiring）：修复 residual 切片：在 MiddleAllReduceRMSNormPattern 替换中使用 tp_rank 感知索引；导入 tp_rank 并更新注释，说明切片在 NoOpElimination 前临时不正确但无害。
- tests/compile/correctness_e2e/test_sequence_parallel.py（模块 序列并行测试；类别 test；类型 test-coverage；符号 test_tp_sp_generation_prompt_embeds）：新增针对 prompt_embeds 的回归测试，覆盖 SP only 和 SP+PP 场景，确保修复的正确性。

关键符号：sync_and_gather_intermediate_tensors, sync_and_slice_intermediate_tensors, MiddleAllReduceRMSNormPattern.replacement, MiddleAllReduceRMSNormStaticFP8Pattern.replacement, _compare_sp, test_tp_sp_generation_prompt_embeds

关键源码片段

vllm/v1/worker/gpu_model_runner.py

核心变更文件：将 sync_and_slice_intermediate_tensors 重构为 sync_and_gather_intermediate_tensors，在 SP 模式下对 residual 进行 all-gather 而非简单切片，确保完整语义。

```
# 变更为 all-gather 方式获取完整 residual（在 PP+SP 场景下游需要完整 tensor）
def sync_and_gather_intermediate_tensors(
    self,
    num_tokens: int,
    intermediate_tensors: IntermediateTensors | None,
    sync_self: bool,
) -> IntermediateTensors:
    assert self.intermediate_tensors is not None
    tp = self.vllm_config.parallel_config.tensor_parallel_size
    is_rs = is_residual_scattered_for_sp(self.vllm_config, num_tokens)

    # 当 sequence parallelism 启用时，“residual” tensor 在 TP rank 间分片
    # 此处通过 all-gather 收集完整 residual，因为下游 QKV + Attention 需要
    # 在 SP 切分点之前的完整 residual。
    if sync_self:
        assert intermediate_tensors is not None
        for k, v in intermediate_tensors.items():
            is_scattered = k == "residual" and is_rs
            if is_scattered:
                local_len = num_tokens // tp
                v = get_tp_group().all_gather(v[:local_len], dim=0)
            # 写入完整 num_tokens
```

```
            self.intermediate_tensors[k][:num_tokens].copy_(
                v[:num_tokens], non_blocking=True
            )

    # 返回时直接使用完整长度（不再根据 is_rs 切片）
    return IntermediateTensors(
        {k: v[:num_tokens] for k, v in self.intermediate_tensors.items()}
    )
```

vllm/compilation/passes/fusion/sequence_parallelism.py

修复 residual 切片：在 MiddleAllReduceRMSNormPattern 替换中使用 tp_rank 感知索引；导入 tp_rank 并更新注释，说明切片在 NoOpElimination 前临时不正确但无害。

```
# 从 _SequenceParallelPatternHelper 基类开始
class _SequenceParallelPatternHelper:
    def __init__(self, epsilon, dtype, device):
        self.epsilon = epsilon
        self.dtype = dtype
        self.device = device
        self.tp_group = get_tp_group()
        self.tp_size = get_tensor_model_parallel_world_size()
        self.tp_rank = get_tensor_model_parallel_rank() # 新增：记录当前 rank
    # ...

# MiddleAllReduceRMSNormPattern 的 replacement 关键差异：
def replacement(
    residual, mm_1, rms_norm_weights
):
    reduce_scatter = self._reduce_scatter(mm_1)
    local_len = reduce_scatter.size(0)
    # 关键修复：使用 tp_rank * local_len 作为起始索引，不再是固定 0
    # 这样每个 rank 正确切片自己的分片
    residual = residual[
        self.tp_rank * local_len : self.tp_rank * local_len + local_len, ...
    ]
    rmsnorm = vllm.ir.ops.fused_add_rms_norm(
        reduce_scatter, residual, rms_norm_weights, self.epsilon
    )
    all_gather = self._all_gather(rmsnorm[0])
    return all_gather, rmsnorm[1]
```

tests/compile/correctness_e2e/test_sequence_parallel.py

新增针对 prompt_embeds 的回归测试，覆盖 SP only 和 SP+PP 场景，确保修复的正确性。

```
# 新增的回归测试函数
SP_PROMPT_EMBEDS_PARALLEL_SETUPS = [
    ParallelSetup(
        tp_size=2,
        pp_size=pp_size,
```

```
        fuse_norm_quant=False,
        fuse_act_quant=False,
        eager_mode=False,
        chunked_prefill=False,
    )
    for pp_size in [1, 2] # 同时覆盖 SP only (pp=1) 和 SP+PP (pp=2)
]

@pytest.mark.parametrize("parallel_setup", SP_PROMPT_EMBEDS_PARALLEL_SETUPS)
@pytest.mark.parametrize("use_inductor_graph_partition", [True, False])
@create_new_process_for_each_test()
def test_tp_sp_generation_prompt_embeds(
    parallel_setup: ParallelSetup,
    num_gpus_available,
    use_inductor_graph_partition: bool,
):
    if use_inductor_graph_partition and not is_torch_equal_or_newer("2.9.0.dev"):
        pytest.skip("inductor graph partition is only available in PyTorch 2.9+")

    _compare_sp(
        "hmellor/tiny-random-LlamaForCausalLM",
        parallel_setup,
        distributed_backend="mp",
        runner="auto",
        test_options=SPTestOptions(multi_node_only=False, load_format=None),
        num_gpus_available=num_gpus_available,
        use_inductor_graph_partition=use_inductor_graph_partition,
        fuse_gemm_comms=False,
        enable_prompt_embeds=True, # 关键：开启 prompt_embeds 模式
        method="generate",
        is_multimodal=False,
    )
```

评论区精华

核心讨论围绕 residual 切片的正确性展开。ProExpertProg 指出原始实现假设编译 pass 是 top-down 替换，但在多模态 /PromptEmbeds 场景中 embedding 层不在 FX graph 内，导致 FirstAllReduceRMSNormPattern 不匹配，MiddleAllReduceRMSNormPattern 接收的是 full-length residual。提出了三种解决方案：

1) 保持当前方法并更新注释； 2) 将切片直接融入模式； 3) 插入 all-gather 并在后续 pass 清理。团队选择方案 1，因为 NoOpEliminationPass 会在同一次 pass 调用后去除无效切片。此外，PP+SP 的兼容性也经过多轮迭代，最终通过 extra all-gather 解决。

- residual 切片三种方案的权衡 (design)：采用方案 1，并更新注释说明临时图语义。后续可能对 MoE SP 实施方案 3。
- InputMutationSlicingPass 的引入与移除 (design)：移除 InputMutationSlicingPass，依赖 #36823 的改进。

- PP+SP 兼容性：all-gather 的正确性 (correctness): 采用 all-gather 方式收集完整 residual，接受额外同步开销。

风险与影响

- 风险：核心风险在于修改了 SP 编译 pass 和 GPU model runner 的数据处理流程（sync_and_gather_intermediate_tensors），可能影响所有使用 SP 的模型。但经过性能与正确性测试（Qwen3-32B 多模态、gsm8k 等），未出现回归。移除 InputMutationSlicingPass 依赖 #36823 的 vllm IR，若该提交未完全 merge 或不同步，可能导致 FP 依然失败。此外，PP+SP 场景新增的 all-gather 会引入额外同步开销，但属于正确性前提。
- 影响：影响范围：启用 SP 的所有用户，特别是多模态模型和 --enable-prompt-embeds 用户。此前 SP 在此类场景完全不可用，修复后可与 SP 配合使用。影响程度：中高（功能修复），但修复经过充分测试，且性能影响微小。团队协作：主要贡献者 wangxingran222 与 reviewer ProExpertProg 进行了多轮讨论，最终达成共识。
- 风险标记：序列并行核心修改，影响多模态和 prompt embeds 路径，依赖 #36823 的 vllm IR, PP+SP 额外 all-gather 通信开销

关联脉络

- PR #36823 Use vllm IR for fused_add_rms_norm: 此 PR 将 fused_add_rms_norm 纳入 vllm IR，消除了 AOT Autograd 插入形状冲突 copy_ 的问题，使得本 PR 可以移除过渡性的 InputMutationSlicingPass，简化实现。
- PR #35771 RFC: Remove piecewise support for SP: ProExpertProg 发起的 RFC，讨论移除 SP 的 piecewise 编译支持（use_inductor_graph_partition=False），本 PR 的讨论中提及该 RFC 以解决 piecewise 模式下 residual 切片不正确的问题。
- PR #33088 Some other SP-related fix (if applicable): Accuracy tests 在 #33088 基础上进行，但该 PR 尚未合并。