PR #30997 完整报告

sgl-project/sglang

[Disagg][Qwen3.5] Fix heterogeneous attn-TP scatter transfer: GDN conv sub-block slice + GQA replicated-KV head map

合并时间：2026-07-16 02:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30997

执行摘要

- 一句话：修复异构 TP 分离下 GDN conv 状态切片和 GQA 头映射错误
- 推荐动作：此 PR 值得仔细阅读，特别是 compute_mamba_state_slice_blocks 的设计方式：通过元数据将"头部独立分片"抽象出来，并统一处理 scatter/aggregation 两种方向。这种模式可复用于其他非线性状态传输。测试用例可选作模型验证的参考。

功能与动机

Qwen3.5 等混合 GDN + GQA 模型在 PD 分离场景中，若预填充和解码使用不同 attention TP 大小（如 DEP4 prefill → TP4 decode），gsm8k 准确率从 0.99 骤降至 0.44。分析发现两个独立 bug：GDN conv_state 在异构 TP 传输时因连续切片导致解码的非 0 秩读取错误的 q/k/v 子块；以及 GQA 中头复制时使用 dst_tp_rank % nkv 而不是 dst_tp_rank // num_kv_head_replicas，导致 replicate 的秩获取错误头（参见 PR#19086 仅修复 gather 方向）。

实现拆解

1. 添加 conv_shard_groups 元数据：在 python/sglang/srt/configs/mamba_utils.py 的 Mamba2StateShape 中添加可选的 conv_shard_groups 字段（GDN 模型为 [key_dim, key_dim, value_dim]），表示 conv_state 子块独立 head-shard 的结构。并在 python/sglang/srt/configs/qwen3_next.py 中为 Qwen3Next/GLM-Image 的重写方法设置该值。
2. 在内存池中暴露 conv_shard_groups：在 python/sglang/srt/mem_cache/memory_pool.py 的 Mamba2TokenToKVPool 中读取并存储 conv_shard_groups（从 cache_params.shape.conv_shard_groups 获取），并新增 get_state_conv_shard_groups() 方法，该方法的返回值与 get_state_dim_per_tensor() 对齐，以便按张量传递子块信息。
3. 核心切片函数：在 python/sglang/srt/disaggregation/utils.py 中新增 compute_mamba_state_slice_blocks() 函数，接受 conv_shard_groups 参数。对于 scatter 方向（预填充 TP < 解码 TP）和 aggregation 方向（预填充 TP > 解码 TP），当 use_subdims 为 True 时，遍历每个子块独立计算 (src_dim_start, dst_dim_start, num_dims) 三元组，避免跨越子块边界。对常规 Mamba2 或 temporal_state 不传入 conv_shard_groups，行为与旧版本（连续切片）相同。

4. 在 mooncake 和 nixl 发送器中集成：分别在 python/sglang/srt/disaggregation/mooncake/conn.py 和 python/sglang/srt/disaggregation/nixl/conn.py 的 _send_mamba_state_slice 中，将 src_state_conv_shard_groups 传入 compute_mamba_state_slice_blocks，替代原来的手动偏移计算。同时修复 GQA 头映射：在 send_kvcache_slice 的 scatter 分支中使用 dst_replication = max(1, dst_attn_tp_size // total_kv_heads) 和 unique_dst_head_idx = dst_tp_rank_in_group // dst_replication，确保连续的解码秩共享同一个 KV 头。
5. 添加回归测试：在 test/registered/disaggregation/test_disaggregation_different_tp.py 中新增 TestDisaggregationGDNHybridHeteroTP 类，配置预填充 TP=1、解码 TP=4 在小型 GDN 混合模型上运行 gsm8k 评估，断言得分大于 0.60。

关键文件：

- python/sglang/srt/disaggregation/utils.py（模块 状态传输；类别 source；类型 core-logic；符号 compute_mamba_state_slice_blocks）：核心函数 compute_mamba_state_slice_blocks 实现 conv_state 子块独立切片的逻辑，同时处理 scatter 和 aggregation 两个方向，是修复的关键。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 get_state_conv_shard_groups）：从 cache_params 中读取并存储 conv_shard_groups，新增 get_state_conv_shard_groups() 方法供状态传输使用，是数据源的管道。
- test/registered/disaggregation/test_disaggregation_different_tp.py（模块 测试；类别 test；类型 test-coverage；符号 TestDisaggregationGDNHybridHeteroTP, setUpClass, start_prefill, start_decode）：新增 TestDisaggregationGDNHybridHeteroTP 回归测试，覆盖 scatter 方向（prefill TP1→decode TP4），直接验证修复效果。
- python/sglang/srt/configs/mamba_utils.py（模块 配置；类别 source；类型 core-logic）：在 Mamba2StateShape 中添加 conv_shard_groups 字段，并修改 create 方法使其可传递，作为元数据入口。
- python/sglang/srt/configs/qwen3_next.py（模块 配置；类别 source；类型 core-logic）：为 Qwen3Next 模型设置正确的 conv_shard_groups 值，使得对该模型生效。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 状态传输；类别 source；类型 dependency-wiring）：集成 compute_mamba_state_slice_blocks 并修复 GQA 头映射的 scatter 分支，是实际传输路径。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 状态传输；类别 source；类型 dependency-wiring）：同样集成 compute_mamba_state_slice_blocks，与 mooncake 并行的传输后端。
- python/sglang/srt/disaggregation/base/conn.py（模块 数据结构；类别 source；类型 core-logic）：在 KVArgs 中添加 state_conv_shard_groups 字段，使数据传递到发送器。
- python/sglang/srt/disaggregation/common/staging_buffer.py（模块 状态传输；类别 source；类型 dependency-wiring）：在 staging handler 中传递 conv_shard_groups，确保 staging 模式也能正确切片。

关键符号：compute_mamba_state_slice_blocks, Mamba2StateShape.create, Mamba2TokenToKVPool.get_state_conv_shard_groups, NixlKVSender._send_mamba_state_slice, MooncakeKVSender._send_mamba_state_slice, MooncakeKVSender.send_kvcache_slice

关键源码片段

python/sglang/srt/disaggregation/utils.py

核心函数 compute_mamba_state_slice_blocks 实现 conv_state 子块独立切片的逻辑，同时处理 scatter 和 aggregation 两个方向，是修复的关键。

```
def compute_mamba_state_slice_blocks(
    src_dim: int,
    dst_dim: int,
    src_attn_tp_size: int,
    dst_attn_tp_size: int,
    dst_tp_rank_in_group: int,
    local_tp_rank_in_group: int,
    conv_shard_groups: Optional[List[int]] = None,
) -> List[Tuple[int, int, int]]:
    """
    计算在异构 attn-TP 之间传输单个 mamba 状态 item 时需要复制的块信息。
    返回 (src_dim_start, dst_dim_start, num_dims) 三元组。

    对于单轴状态（temporal_state 或非 GDN），conv_shard_groups 为 None，
    返回单个连续块，与旧行为字节一致。

    对于 GDN conv_state（cat([query, key, value])），每个子块独立 head-shard，
    因此需要按子块切片，避免跨越边界。
    """
    use_subdims = (
        conv_shard_groups is not None
        and sum(conv_shard_groups) == src_dim * src_attn_tp_size
    )

    if src_attn_tp_size > dst_attn_tp_size:
        # Aggregation 方向：多个 prefill 秩向一个 decode 秩写入
        writers_per_decode = src_attn_tp_size // dst_attn_tp_size
        local_writer_idx = local_tp_rank_in_group % writers_per_decode
        if not use_subdims:
            return [(0, local_writer_idx * src_dim, src_dim)]
        # 对于 conv_state：按子块放置，使 decode 缓冲区为 [q0,q1,...,k0,k1,...,v0,v1,...]
        blocks: List[Tuple[int, int, int]] = []
        src_off = 0
        dst_off = 0
        for full_sd in conv_shard_groups:
            src_sub = full_sd // src_attn_tp_size
            dst_sub = full_sd // dst_attn_tp_size
```

```
            blocks.append((src_off, dst_off + local_writer_idx * src_sub, src_sub))
            src_off += src_sub
            dst_off += dst_sub
        return blocks

    # Scatter 方向：1 个 prefill 秩向多个 decode 秩发送
    if not use_subdims:
        src_dim_start = (dst_tp_rank_in_group * dst_dim) % src_dim
        return [(src_dim_start, 0, dst_dim)]

    # conv_state：从三个独立 head-shard 的子块中收集 decode 秩的 [q, k, v]
    blocks: List[Tuple[int, int, int]] = []
    src_off = 0
    dst_off = 0
    for full_sd in conv_shard_groups:
        src_sub = full_sd // src_attn_tp_size
        dst_sub = full_sd // dst_attn_tp_size
        src_start = src_off + (dst_tp_rank_in_group * dst_sub) % src_sub
        blocks.append((src_start, dst_off, dst_sub))
        src_off += src_sub
        dst_off += dst_sub
    return blocks
```

python/sglang/srt/mem_cache/memory_pool.py

从 cache_params 中读取并存储 conv_shard_groups，新增 get_state_conv_shard_groups() 方法供状态传输使用，是数据源的管道。

```
class Mamba2TokenToKVPool:
    def __init__(self, ...):
        ...
        # Full (unsharded) conv sub-block dims for PD transfer across different
        # attn_tp_size (GDN: [key_dim, key_dim, value_dim]); None otherwise.
        self.conv_shard_groups = getattr(cache_params.shape, "conv_shard_groups", None)

    def get_state_conv_shard_groups(self):
        """
        返回按张量对齐的 conv 子块维度列表，与 get_state_dim_per_tensor() 对齐。
        对于 conv_state 且 conv_shard_groups 非 None，返回完整子块维度；
        其他情况返回 None。
        """
        subdims_per_tensor = []
        for field in vars(self.mamba_cache):
            if field in (
                "intermediate_ssm",
                "intermediate_conv_window",
                "replayssm_d",
                "replayssm_k",
                "replayssm_g",
            ):
```

```
            continue
        value = getattr(self.mamba_cache, field)
        if value is None:
            continue
        tensors = value if isinstance(value, list) else [value]
        for _ in tensors:
            subdims = (
                list(self.conv_shard_groups)
                if field == "conv" and self.conv_shard_groups is not None
                else None
            )
            subdims_per_tensor += [subdims] * self.num_mamba_layers
    return subdims_per_tensor
```

评论区精华

#3568925621(gemini-code-assist[bot]): 指出 aggregation 方向（src_attn_tp_size > dst_attn_tp_size）忽略 conv_shard_groups，导致目标缓冲区中子块交错（[q0,k0,v0,q1,...]）而非连续排列（[q0,q1,...,k0,k1,...,v0,v1,...]）。 YAMY1234(回复): 已通过 commit d64c4942a7 修复，在 aggregation 方向也正确遍历子块。 ShangmingCai(Approval): "The PD Disaggregation part looks good."

- Aggregation 方向 conv_shard_groups 处理缺失 (design): YAMY1234 在 commit d64c4942a7 中修复了该问题。

风险与影响

- 风险：
  - 回归风险：核心切片逻辑变更可能影响非 GDN 模型（如标准 Mamba2），但 conv_shard_groups 显式声明且默认 None，保持单连续切片行为（字节一致）。头映射修改仅在 total_kv_heads < decode attn_tp 时触发，其他场景行为不变。
  - 性能风险：对 conv_state 每层计算多个 (src_start, dst_start, num_dims) 三元组并发送多个 RDMA 段，但子块数量固定（GDN 为 3），开销可忽略。
  - 兼容性：要求发送器（mooncake/nixl）的 bootstrap 阶段传递 state_conv_shard_groups，旧版本解码端忽略该字段（默认 None）仍可正常工作。
  - 测试覆盖：新增测试仅覆盖 scatter 方向（prefill TP=1 → decode TP=4），未覆盖 aggregation 方向（prefill TP=4 → decode TP=1）的组合。
- 影响：
  - 用户影响：使用 Qwen3.5 等混合 GDN + GQA 模型且开启 PD 分离的用户将直接从精度恢复中受益（gsm8k 从 0.44 升至 0.9756）。其他模型（纯注意力或无 GQA 复制）不受影响。
  - 系统影响：修改了 PD 状态传输的核心路径，但通过条件保护保证向后兼容。
  - 团队影响：合并者需确认 CI 通过；审阅者已 approve。
  - 风险标记：核心状态传输路径变更，可能影响 Qwen3.5 精度，需要验证非 GDN 模型的正确性

关联脉络

- PR #23744 [Bugfix] Fix heterogeneous TP disaggregation precision for GQA head mapping and Mamba conv state: 同一个问题的早期修复尝试，但未合并；本 PR 在 refactored 代码上重新实现并扩展了修复。
- PR #19086 [Disagg] Fix GQA head mapping for gather direction: 本 PR 补充了上次没有覆盖的 scatter 方向的 GQA 头映射修复。