

PR #50005 完整报告

vllm-project/vllm

[Bugfix][DCP] Fix NVIDIA DeepSeek-V3.2 / GLM-5.2 fused attention

合并时间: 2026-08-31 15:30

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/50005>

执行摘要

- 一句话: 修复 DSV3.2/GLM-5.2 融合注意力在纯 DCP 下的两处正确性缺陷
- 推荐动作: 值得精读。该 PR 展示了跨 PR 协调的正确范式: 先用最小改动止血、再随依赖 PR 落地切换到集中化实现, 避免重复实现; 配合实验证据 (DCP4 vs DCP1 byte-identical) 验证正确性, 而非仅靠单元测试。重点关注 MLADCPManager 作为集中式通信控制器的设计取舍, 以及 kernel 层面

功能与动机

关联 Issue #50095 明确指出融合路径存在两个纯 DCP 正确性缺陷: (1)

`vllm/models/deepseek_v32/common/kernels.py` 在 owner-local KV slot 有效性检查之前就返回, 负 slot 意味着另一个 DCP rank 拥有该 token 的 KV 条目, 并不消除本 rank 的 query 贡献; (2) `vllm/models/deepseek_v32/attention.py` 仅在 PCP+DCP 下收集 query heads 并合并 partial output/LSE, 纯 DCP 因此用 rank-local query heads 对 owner-local KV 做注意力, 且从不执行跨 rank 结果合并。可观测回归是 unpatched GLM-5.2-NVFP4 在 TP4/DCP4 + `fp8_ds_mla` 下输出重复 token 垃圾 (如 `locklocklock...`), 而 patched 路径在贪婪解码下与 DCP1 逐字节一致。

实现拆解

实现分为四个步骤:

1. Triton 融合 kernel 调整 (`vllm/models/deepseek_v32/common/kernels.py`): 将 `pid == 2` 分支的 Q RMSNorm 移到 slot_mapping 负值检查之前, 使其对每一行无条件执行; slot 检查只保留在 K 侧 (`pid == 1`) 之前, 用于抑制 MLA 与 indexer KV cache 写入。原因是 DCP 下 query 不分片, 每个 rank 都需要完整 query 集合参与后续 attention, 负 slot 只表示 KV 槽位归其他 rank 所有。注释明确说明 padding 行多做一次 row-local 计算是无害的。
2. pure DCP 的 query heads 收集 (`vllm/models/deepseek_v32/attention.py` 的 `_sparse_indexer_and_attn`): 在原有 PCP all-gather 分支后新增 `elif not self.use_pcp and self.impl.dcp_world_size > 1` 分支, 将 `(ql_nope, mqa_q)` 拼接后通过 `self.dcp_manager.query_gather` 收集。该写法是 review 迭代的结果: 最初硬编码 `dcp_manager.group.all_gather`, 在 #52377 落地后切换为 `query_gather`, 从而保留 decode-only batch 的 direct-workspace 快路径, 并把 oversized mixed batch 的 fallback 交给 `MLADCPManager` 统一处理。

3. partial output/LSE 跨 rank 合并条件放宽: combine 的条件从 `self.use_pcp and self.impl.dcp_world_size > 1` 改为 `self.impl.dcp_world_size > 1`, 使 pure DCP 也执行 `dcp_manager.combine`。PCP 分支保留原有的 `seq_lens / query_start_loc` 元数据计算; pure DCP 分支传 `None`, 因为稀疏后端对空 local shard 输出 `(0, -inf)` 中性贡献, 不需要 PCP-only 的空分片元数据。`finalize_mla_pcp_decode` 仅在 `use_pcp` 时调用。
4. 回归测试 (`tests/kernels/test_fused_deepseek_v32_norm_rope.py`): 新增 `test_fused_norm_rope_normalizes_query_without_local_cache_slots`, 构造全负 slot 映射 (模拟不拥有任何 KV 槽位的 DCP rank), 断言 query 输出与参考 RMSNorm 一致、MLA 与 indexer cache 未被写入、Top-K 缓冲被清为 -1。该测试在未打补丁的 main 上是唯一失败的用例, 证明其精确捕获了缺陷。

配套变更还包括 8 个 commit 的演进: drakosha 在 8 月文件迁移 (`deepseek_v32/nvidia/` → `deepseek_v32/`) 后做了 forward-port 合并, 保证分支可干净合并到 main。

关键文件:

- `vllm/models/deepseek_v32/attention.py` (模块 注意力层; 类别 source; 类型 core-logic; 符号 `_sparse_indexer_and_attn`): 修复的核心文件: 新增 pure DCP 分支通过 `dcp_manager.query_gather` 收集 query heads, 并将 partial output/LSE 的 combine 条件从 PCP-only 放宽到所有 DCP 场景。
- `vllm/models/deepseek_v32/common/kernels.py` (模块 融合内核; 类别 source; 类型 core-logic; 符号 `_fused_norm_rope_kernel`): 融合 norm/RoPE Triton kernel 的 Q RMSNorm 从 slot 检查之后移到之前, 消除 DCP 非 owner rank 上 query 输出为零的缺陷。
- `tests/kernels/test_fused_deepseek_v32_norm_rope.py` (模块 内核测试; 类别 test; 类型 test-coverage; 符号 `test_fused_norm_rope_normalizes_query_without_local_cache_slots`): 新增全负 slot 回归测试, 精确复现 DCP 非 owner rank 场景, 未打补丁时是唯一失败用例, 验证了修复的有效性。

关键符号: `_sparse_indexer_and_attn`, `_fused_norm_rope_kernel`,
`test_fused_norm_rope_normalizes_query_without_local_cache_slots`

关键源码片段

`vllm/models/deepseek_v32/attention.py`

修复的核心文件: 新增 pure DCP 分支通过 `dcp_manager.query_gather` 收集 query heads, 并将 partial output/LSE 的 combine 条件从 PCP-only 放宽到所有 DCP 场景。

```
if self.use_pcp and self.impl.dcp_world_size > self.impl.pcp_world_size:
    # PCP+DCP: 先做 TP all-gather, 保证每个 rank 持有完整 query head 集合
    if isinstance(mqa_q_arg, tuple):
        mqa_q_arg = torch.cat(mqa_q_arg, dim=-1)
        mqa_q_arg = get_tp_group().all_gather(mqa_q_arg, dim=1)
    elif not self.use_pcp and self.impl.dcp_world_size > 1:
        # Pure DCP: query 不分片, 每个 rank 都需要完整 head 集参与
        # owner-local KV 的注意力。走 MLADCPManager.query_gather 而非裸
        # all_gather: decode-only batch 保留 direct-workspace 快路径,
```

```

# 超长 prefill 的混合 batch 由 manager 回退到 allgather
assert self.dcp_manager is not None
if isinstance(mqa_q_arg, tuple):
    mqa_q_arg = torch.cat(mqa_q_arg, dim=-1)
assert self.dcp_manager.query_gather is not None
mqa_q_arg = self.dcp_manager.query_gather(mqa_q_arg)
attn_out, lse = self.impl.forward_mqa( # type: ignore[attr-defined]
    mqa_q_arg, kv_cache, attn_metadata, self
)

if self.impl.dcp_world_size > 1:
    # 只要有 DCP 就必须跨 rank 合并 partial output 与 LSE;
    # 修复前该合并只在 PCP+DCP 下执行, pure DCP 输出是残次品
    assert lse is not None and self.dcp_manager is not None
    seq_lens: torch.Tensor | None
    query_start_loc: torch.Tensor | None
    if self.use_pcp:
        if attn_metadata.decode is not None:
            seq_lens = attn_metadata.decode.seq_lens
        else:
            all_seq_lens = cast(
                torch.Tensor,
                attn_metadata.seq_lens, # type: ignore[attr-defined]
            )
            seq_lens = all_seq_lens[: attn_metadata.num_decodes]
            query_start_loc = attn_metadata.query_start_loc[
                : attn_metadata.num_decodes + 1
            ]
        else:
            # Pure DCP: 稀疏后端对空 local shard 输出 (0, -inf)
            # 中性贡献, 无需 PCP-only 的空分片元数据
            seq_lens = None
            query_start_loc = None
    attn_out = self.dcp_manager.combine(
        attn_out,
        lse,
        seq_lens=seq_lens, # type: ignore[arg-type]
        query_start_loc=query_start_loc, # type: ignore[arg-type]
    )
    if self.use_pcp:
        attn_out = finalize_mla_pcp_decode(attn_out, self.num_heads)

```

[vllm/models/deepseek_v32/common/kernels.py](#)

融合 norm/RoPE Triton kernel 的 Q RMSNorm 从 slot 检查之后移到之前, 消除 DCP 非 owner rank 上 query 输出为零的缺陷。

```

tok_idx = tl.program_id(0).to(tl.int64)
pid = tl.program_id(1)
if USE_PDL:

```

```

tl.extra.cuda.gdc_wait()
tl.extra.cuda.gdc_launch_dependents()

# pid == 3: indexer 层将 Top-K 缓冲置 -1；共享层直接复用上一层结果
if pid == 3:
    if not HAS_INDEXER:
        return
    for i in range(0, TOPK, TOPK_BLOCK_SIZE):
        offset = i + tl.arange(0, TOPK_BLOCK_SIZE)
        mask = offset < TOPK
        tl.store(
            topk_indices_ptr + tok_idx * topk_indices_stride + offset,
            -1,
            mask=mask,
        )
    return

if pid == 2:
    # Q RMSNorm 对每一行无条件执行：DCP 下负 slot 只表示该 token 的
    # KV 槽位归其他 rank 所有，query 本身在每个 rank 上都需要
    # （query 不分片），所以 slot 检查不能 gate 它。Padding 行只是
    # 多做一次 row-local 计算，无位置加载、无 cache 写入，开销可忽略
    q_block = tl.arange(0, Q_BLOCK_SIZE)
    q_mask = q_block < Q_DIM
    q_c = tl.load(q_c_ptr + tok_idx * q_c_stride + q_block, mask=q_mask, other=0.0)
    q_c_rms_w = tl.load(q_rms_norm_w_ptr + q_block, mask=q_mask)
    q_c = _rms_norm(q_c, q_c_rms_w, q_rms_eps, Q_DIM)
    tl.store(q_c_out_ptr + tok_idx * q_c_out_stride + q_block, q_c, mask=q_mask)
    return

if slot_mapping_ptr is None:
    if kv_out_ptr is None and kpe_out_ptr is None and index_k_out_ptr is None:
        return
    elif tl.load(slot_mapping_ptr + tok_idx) < 0:
        # Padding，或 DCP 下 KV 槽位归其他 rank 所有：
        # K 侧 norm 与 MLA/indexer cache 写入被抑制
        return

if pid == 1:
    # KV RMSNorm + KV RoPE + MLA concat_and_cache 融合：
    # normed kv_c 与 RoPE 后的 k_pe 直接在寄存器中写入 MLA KV cache，
    # 省去一次独立 kernel 的读写往返
    kv_block = tl.arange(0, KV_DIM)
    kv_c = tl.load(kv_ptr + tok_idx * kv_stride + kv_block)
    kv_c_rms_w = tl.load(kv_rms_norm_w_ptr + kv_block)
    kv_c = _rms_norm(kv_c, kv_c_rms_w, kv_rms_eps, KV_DIM)
    # ... 后续为 KV RoPE 与 cache 写入（见原文件）

```

评论区精华

Review 中最有价值的交锋集中在 query 收集方式与跨 PR 分工上：

- cjackal 质疑硬编码 all_gather：认为 DCP Manager 是模型引擎生命周期内集中式通信控制器，#52377 的 q-gather / combine fallback 应作为首选，硬编码 allgather 到模型脚本不利于长期维护；同时指出 #52377 还能保留 pure-decode batch 的 direct workspace 快路径。
- drakosha 用实验证据澄清两个 PR 不可互换：在 4xH200 GLM-5.2-NVFP4 TP4/DCP4 上，nightly + #52377 单独仍输出重复 token 垃圾，nightly + 本 PR 则与 DCP1 逐字节一致。原因是从 #52861 起这两个架构在 NVIDIA 上走 vllm/models/deepseek_v32/ 路由，#52377 修复的调用点已不在该路径执行；#52377 仍覆盖 ROCm 的 compiled 路由，所以两边都需要。
- LucasWilkinson 决定先合并 #52377 再更新本 PR：认可集中化方向后，将本 PR 的 query gather 暂时保留为裸 all_gather 并加 TODO，待 #52377 落地后切换。foraxe 在 22e9a19f8d 完成了切换并删除 TODO。
- LucasWilkinson 的 style nit：要求精简 pure-DCP 注释并补充 TODO，foraxe 在 6127a179f8 处理，将注释收敛为两行 empty-shard 不变量描述。
 - query 收集应走 dcp_manager.query_gather 而非硬编码 all_gather (design)：#52377 落地后，foraxe 在 22e9a19f8d 将调用切换为 dcp_manager.query_gather 并删除 TODO，线程关闭。
 - 52377 与本 PR 是否可互换 (correctness)：两个 PR 各管一条路径，均需保留：#52377 覆盖通用 /ROCm 路径，本 PR 覆盖 NVIDIA 融合路径。
 - 评论精简与 TODO 补充 (style)：foraxe 在 6127a179f8 精简注释为两行 empty-shard 不变量描述；TODO 在 #52377 落地后随切换一并移除。
 - 文件迁移后的 forward-port (other)：合并 forward-port 后分支可干净合并，且只包含预期的三个文件。

风险与影响

- 风险：主要风险集中在三处：
 1. 核心 attention 路径变更：_sparse_indexer_and_attn 是 DSV3.2/GLM-5.2 的注意力主路径，combine 条件从 use_pcp and dcp_world_size > 1 放宽为 dcp_world_size > 1。PCP 分支的 seq_lens / query_start_loc 计算逻辑与原版等价（仅作变量类型标注调整），且 finalize_mla_pcp_decode 仍限定在 use_pcp 内，PCP 行为理论上不变，但缺少针对 PCP+DCP 组合的显式回归验证。
 2. 对 #52377 的依赖：dcp_manager.query_gather 对 oversized mixed batch 的正确性依赖 MLADCPManager 的 allgather fallback；若该 fallback 存在边界缺陷，会在超长 prefill + 稀疏 MLA 混合批次上暴露。
 3. kernel 无条件 Q RMSNorm 的边际开销：padding 行现在也会执行 Q RMSNorm 计算，虽无副作用但有少量额外算力消耗，在高 padding 率场景可能引入轻微性能抖动。多卡 DCP 行为 (byte-identical 对比) 依赖 drakosha 的手动 H200 验证，未固化进 CI，存在回归漏检窗口。 - 影响：影响范围限定在使用 NVIDIA DeepSeek-V3.2 / GLM-5.2 融合注

意力路径且开启 DCP (`dcp_world_size > 1`) 的部署：修复前纯 DCP 会输出重复 token 垃圾，修复后与 DCP1 输出逐字节一致。DCP=1 的单卡部署无行为变化（kernel 仅多执行 padding 行的 Q RMSNorm）。对团队而言，这是 DCP 功能线的重要正确性修复，同时确立了 NVIDIA 稀疏 MLA 路径与通用路径（ROCm compiled route）各自依赖 `deepseek_v32/` 与通用 `sparse-MLA` 后端的双轨格局。 - 风险标记：核心 attention 路径变更，多卡行为依赖手动验证，跨 PR 依赖 #52377

关联脉络

- PR #52377 Generic sparse-MLA forced-MQA metadata and direct-workspace fallback（标题据讨论描述归纳）：本 PR 的 `query_gather` 最终切换为 `dcp_manager.query_gather`，依赖 #52377 提供的 `oversize fallback` 与 `direct-workspace` 快路径；讨论中确认两者各覆盖 NVIDIA 融合路径与通用 /ROCm 路径。
- PR #52861 MRV2 forward override for DSV3.2/GLM-5.2 routing on NVIDIA（标题据讨论描述归纳）：drakosha 在讨论中指出自 #52861 起这两个架构在 NVIDIA 上走 `deepseek_v32/` 路由，这是 #52377 无法覆盖本路径、必须单独修复的直接原因。
- PR #50009 Shared-DCP performance work（标题据 PR body 描述归纳）：PR body 明确说明本正确性修复独立于 Shared-DCP 性能特性，避免评审者将两者混淆。