PR #34161 完整报告

sgl-project/sglang

fix: preserve GQA head mapping in Triton DCP prefill

合并时间：2026-08-10 09:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34161

执行摘要

- 一句话：修复 DCP prefill 的 GQA 映射错乱，GSM8K 恢复至 0.95
- 推荐动作：值得精读。重点关注 _forward_extend_dcp 中 K/V 切片公式的简洁推导（rank_in_group * tp_k_head_num // world_size 配合 max(..., start + 1) 防空切片），以及 CPU 回归测试用 FakeDcpGroup + fake extend_attention_fwd 固定"无通信、形状正确、选区正确"三个契约的轻量测试模式。该 PR 也提供了一个很好的静默正确性回归修复范例：作者用精确的精度数据（0.77 vs 0.95）证明 bug、用 DCP1 对照证明无回归、用 TP8/DCP2 扩展拓扑覆盖。

功能与动机

PR body 明确指出：After #32858, K/V heads are replicated within a DCP group while Q heads remain sharded，而 Triton DCP current-chunk prefill 路径"passed each local Q shard together with all replicated K/V heads to the generic extend kernel"，使 kernel 在每个 rank 上推断出错误的本地 GQA 映射（TP4/DCP4 下 8 个本地 Q head 对 2 个复制 KV head 被误判为 4:1，真实全局映射为 32:2 即 16:1，ranks 0-1 属于 KV head 0、ranks 2-3 属于 KV head 1）。后果是 This corrupted every fresh prompt's prefill and reduced GSM8K accuracy from the expected ~0.95 to 0.75-0.77。该问题由 #34133 暴露（首次完整跑完 4xB200 流程），属于静默正确性回归——不崩溃但输出错误答案。

实现拆解

1. 定位变更入口：修改集中在 python/sglang/srt/layers/attention/triton_backend.py 的 TritonAttnBackend._forward_extend_dcp，该函数处理 DCP prefill 时 current-chunk 的注意力计算。原实现把 DCP group 内复制的完整 K/V 直通 generic extend kernel，kernel 在每个 rank 上从 KV head 0 重新推导本地 GQA 映射，导致所有新 prompt 的 prefill 静默错误。
2. 按 rank 切分复制的 K/V：在 if k.numel() > 0 分支内、调用 extend_attention_fwd 之前，当 layer.tp_k_head_num > 1（GQA 场景）时，用 kv_head_start = group.rank_in_group * layer.tp_k_head_num // group.world_size 与 kv_head_end = max((group.rank_in_group + 1) * layer.tp_k_head_num // group.world_size, kv_head_start + 1) 计算连续 K/V 区间，并对 k、v 做切片；tp_k_head_num == 1 的 MQA/MLA 场景保持原路径。该方案不引入任何 collective 通信，与既有 prefix-attention（Q all-gather 后按 LSE 合并）路径互不影响。

3. 新增 CPU 回归测试：test/registered/dcp/test_dcp_layout_unit.py 新增 test_gqa_current_chunk_selects_kv_for_the_global_dcp_head_layout，用 FakeDcpGroup（world_size=4、rank_in_group=1）与 fake extend_attention_fwd 纯 CPU 验证三点：不触发 all_gather、传给 kernel 的 Q 形状保持 [1, 2, 2]、rank 1 选中 k[:, 0:1]（即 KV head 0，符合"ranks 0-1 共享 KV head 0"的全局布局）。
4. 精度与性能验证：Qwen3.5-397B-A17B-FP8 + Triton attention + --disable-radix-cache 全量 1,314 例 GSM8K：DCP4 从 0.772/0.751 恢复到 0.954，普通 TP4/DCP1 为 0.950（确认非 DCP 路径无回归），TP8/DCP2（8×H200）为 0.960；DCP4 输出吞吐 1,133.58 tok/s，与"无新增通信、只收窄 K/V 切片"的设计一致。
5. Review 演进：作者自审后移除了 e2e 测试中与 bug 无关的 --watchdog-timeout 1200 残留，并将最初的"全局 Q-head 区间反推 K/V 区间"实现简化为"由 DCP rank 与本地 KV head 数直接计算连续 bucket"，最终提交 refactor: simplify DCP KV head selection。

关键文件：

- python/sglang/srt/layers/attention/triton_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 _forward_extend_dcp）：核心修复点：_forward_extend_dcp 在调用 Triton extend kernel 前按 DCP rank 切出与本地 Q shard 对应的连续 K/V 子集，是修复 GQA 全局映射错乱的关键逻辑。
- test/registered/dcp/test_dcp_layout_unit.py（模块 DCP 测试；类别 test；类型 test-coverage；符号 test_gqa_current_chunk_selects_kv_for_the_global_dcp_head_layout, FakeDcpGroup）：新增 CPU 回归测试锁定 TP4/DCP4 全局 GQA 映射行为，验证无 all_gather、Q 形状与 K/V 选区正确，是防止该静默回归再次引入的测试锚点。

关键符号：_forward_extend_dcp, test_gqa_current_chunk_selects_kv_for_the_global_dcp_head_layout

关键源码片段

python/sglang/srt/layers/attention/triton_backend.py

核心修复点：_forward_extend_dcp 在调用 Triton extend kernel 前按 DCP rank 切出与本地 Q shard 对应的连续 K/V 子集，是修复 GQA 全局映射错乱的关键逻辑。

```
# TritonAttnBackend._forward_extend_dcp 的 current-chunk 阶段（节选）
# 背景：在 #32858 之后 DCP group 内 K/V heads 复制、Q heads 分片；若把完整复制
# 的 K/V 直接交给 generic extend kernel，kernel 会在每个 rank 上从 KV head 0
# 重新推导本地 GQA 映射（TP4/DCP4 下 8 个本地 Q head 对 2 个复制 KV head 被
# 误判为 4:1，而全局应为 32:2 即 16:1），导致新 prompt 的 prefill 静默错误。

group = get_parallel().dcp_group # 提供 rank_in_group 与 world_size
q_local = q.view(-1, layer.tp_q_head_num, layer.qk_head_dim).contiguous()
total_tokens, local_heads, _ = q_local.shape
# current_out / current_lse 的初始化省略；下面进入 current-chunk K/V 处理。

if k.numel() > 0:
    # 仅 GQA（本地有多个 KV head）需要按 rank 重映射；单 KV head 场景保持原行为。
    if layer.tp_k_head_num > 1:
```

```
            # 连续 K/V bucket：world_size 个 rank 均匀消费 tp_k_head_num 个复制的
            # KV head；max(..., kv_head_start + 1) 保底选 1 个 head，防止整除边界
            # 出现空切片。例如 rank 1（TP4/DCP4）会选中 KV head 0，与全局布局一致。
            kv_head_start = (
                group.rank_in_group * layer.tp_k_head_num // group.world_size
            )
            kv_head_end = max(
                (group.rank_in_group + 1) * layer.tp_k_head_num // group.world_size,
                kv_head_start + 1,
            )
            k = k[:, kv_head_start:kv_head_end]
            v = v[:, kv_head_start:kv_head_end]

        # 当前 chunk 的 K/V 在 masked cache write 前仍是本地的，可复用原 extend
        # kernel 的 current-token 阶段（skip_prefix=True，只算当前 token 注意力）。
        empty_kv_indptr = torch.zeros_like(kv_indptr)
        self.extend_attention_fwd(
            q_local,
            k.contiguous(),
            v.contiguous(),
            current_out,
            k_buffer,
            v_buffer,
            self.forward_metadata.qo_indptr,
            empty_kv_indptr,
            kv_indices[:0],
            None,
            causal,
            None,
            max_extend_len,
            1.0,
            1.0,
            sm_scale=layer.scaling,
            logit_cap=logits_soft_cap,
            xai_temperature_len=layer.xai_temperature_len,
            lse_extend=current_lse,
            skip_prefix=True,
        )
```

test/registered/dcp/test_dcp_layout_unit.py

新增 CPU 回归测试锁定 TP4/DCP4 全局 GQA 映射行为，验证无 all_gather、Q 形状与 K/V 选区正确，是防止该静默回归再次引入的测试锚点。

```
# test/registered/dcp/test_dcp_layout_unit.py 新增的 CPU 回归测试（节选）
# 模拟 TP4/DCP4 简化场景：world_size=4、rank_in_group=1、本地 2 个 Q head
# 与 2 个复制 KV head。目标是验证 current-chunk 阶段按全局 GQA 布局选 K/V，
# 且不引入任何 all_gather 通信。

class FakeDcpGroup:
```

```
    world_size = 4
    rank_in_group = 1

    def __init__(self):
        self.all_gather_calls = 0

    def all_gather(self, tensor, dim):
        # 计数并返回"复制后"的张量，用于断言修复路径不触发 collectives。
        self.all_gather_calls += 1
        return torch.cat((tensor, tensor + 10), dim=dim)

group = FakeDcpGroup()
backend = TritonAttnBackend.__new__(TritonAttnBackend) # 绕过 __init__ 纯 CPU 构造

kernel_q_shapes, kernel_k = [], []

def fake_extend_attention(q, k, _v, out, *_args, lse_extend, **_kwargs):
    # 捕获传给真实 kernel 的 Q 形状与 K 内容，同时用写入 out 模拟前向。
    kernel_q_shapes.append(q.shape)
    kernel_k.append(k.clone())
    out.copy_(q.float())
    lse_extend.zero_()

backend.extend_attention_fwd = fake_extend_attention
# layer：tp_q_head_num=2、tp_k_head_num=2、head_dim=2，其余字段仅需占位。

with rc.get_parallel().override(dcp_group=group):
    out = backend._forward_extend_dcp(
        q=q, k=k, v=k.clone(), layer=layer,
        forward_batch=SimpleNamespace(), causal=True,
        logits_soft_cap=0.0, sinks=None,
    )

# 三点关键断言：无 all_gather；Q 保持本地分片形状 [1, 2, 2]；
# rank 1 选中 KV head 0（TP4/DCP4 下 rank 0/1 共享 head 0），而非从头推导 4:1 映射。
self.assertEqual(group.all_gather_calls, 0)
self.assertEqual(kernel_q_shapes, [torch.Size([1, 2, 2])])
self.assertTrue(torch.equal(kernel_k[0], k[:, 0:1]))
self.assertTrue(torch.equal(out, q))
```

评论区精华

本 PR 无外部 reviewer，6 条 review 评论全部来自作者自己的严格自审，三个线程均 resolved：

- 复杂度自审：Logic inside \if k.numel() > 0:` seems overly complicated—— 初始实现先由local_heads * world_size推算全局 Q head 数、再经q_heads_per_kv换算 K/V 区间，并带整除assert；作者随后提交refactor: simplify DCP KV head selection，简化为由rank_in_group、tp_k_head_num、world_size直接计算连续 bucket，并保留max(...,

kv_head_start + 1)` 防空切片。

- 调试残留清理：对 *e2e* 测试中 --watchdog-timeout 1200 给出 This is not needed，随即移除，PR 保持最小改动。
- 注释精简：Trim this comment 后缩为一行 Select the replicated K/V heads matching this rank's Q shard.，详细推导语义保留在回归测试 docstring 与 PR body。
- 此外 issue 评论补充 TP8/DCP2（8×H200）全量验证：GSM8K 0.959665、无推理挂起，确认修复在更大张并行拓扑上同样有效。
- current-chunk K/V 选择逻辑过度复杂 (design): 作者提交 commit refactor: simplify DCP KV head selection，改为直接用 rank_in_group * tp_k_head_num // world_size 计算连续 K/V bucket，并保留 max(..., start + 1) 防空切片。
- 测试中 watchdog-timeout 参数不需要 (style): 已从最终变更中移除该参数，PR 保持只含源码修复与回归测试两个文件的最小改动。
- 注释精简 (style): 已精简为一行 Select the replicated K/V heads matching this rank's Q shard.，关键语义保留在 commit message 与回归测试 docstring 中。

风险与影响

- 风险：
  - 非整除 GQA 场景未覆盖：kv_head_start/end 公式假定 tp_k_head_num 可被 world_size 均匀消费；当 tp_k_head_num % world_size != 0 时，max(..., kv_head_start + 1) 保底选 1 个 head，但全局 Q→KV 映射本身不均匀时切片语义可能与真实布局不完全一致。测试与验证仅覆盖 TP4/DCP4、TP8/DCP2 等可整除场景，非整除配置（如 3 个 KV head 分给 4 个 rank）未纳入。
  - backend 覆盖范围有限：修复只作用于 Triton backend 的 current-chunk 阶段，prefix-attention 与 kernel 内部 GQA 推导逻辑均未改动；其他 backend（如 FA4）若存在 DCP + GQA 组合需要独立排查，本 PR 未声明。
  - 数值验证依赖 *e2e*：CPU 回归测试只验证形状与选区、不验证数值正确性；精度结论依赖 test/registered/dcp/test_qwen3p5_triton_dcp.py（阈值 0.90）与作者提供的实测数据，若 CI 未覆盖该 GPU 套件则回归可能漏检。
  - 属于静默错误类缺陷（不崩溃只出错误答案），容易在发布流程中被遗漏，建议确认修复进入后续 release 分支。
- 影响：
  - 用户影响：使用 Triton DCP prefill + GQA 模型（如 Qwen3.5-397B TP4/DCP4）的部署，新 prompt 的 prefill 从静默错误变为正确，GSM8K 从 0.75-0.77 恢复到 0.95。
  - 性能影响：无新增通信，仅收窄 K/V 切片；DCP4 输出吞吐 1,133.58 tok/s，DCP1 为 1,704.64 tok/s，与“不打乱原有 prefix-attention 路径”的设计一致，未观察到性能回退。
  - 团队影响：test_dcp_layout_unit.py 成为 DCP 布局语义的回归锚点；本修复与 #34133（DCP 拓扑从 ParallelState 派生）形成闭环，为后续 DCP 路径重构提供了行为基准，也降低了类似静默映射错误再次引入的风险。
  - 风险标记：核心路径变更，非整除 GQA 场景未覆盖，静默正确性回归

关联脉络

- PR #34133 config: derive the runner's DCP topology from its ParallelState: PR body 明确本 bug 由 #34133 暴露：DCP 拓扑从 ParallelState 派生后 Triton DCP prefill 路径首次完整跑完（此前 4xB200 运行 cold startup 超时），暴露 0.75-0.77 的 GSM8K 精度回归。
- PR #34096 config: the KV-cache configurator reads the bags: 同一测试文件 test/registered/dcp/test_dcp_layout_unit.py 在 #34096 中被修改，本 PR 继续在该文件新增 GQA 映射的 CPU 回归测试；该文件正成为 DCP 布局行为的中心化回归测试集。