PR #43673 完整报告

vllm-project/vllm

[ROCm][Perf] DSv3.2: fuse MLA Q concat+fp8-quant in forward_mqa

合并时间：2026-06-23 18:12

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43673

执行摘要

- 一句话：融合 MLA Q 拼接与 FP8 量化，减少 ROCm decode 延迟
- 推荐动作：该 PR 值得快速合入，性能效果明确且验证充分。但建议关注 Gemini 的架构建议，在后续清理 PR 中考虑使用 supports_quant_query_input 统一模式，以减少手动调用和潜在不一致。

功能与动机

On the ROCm sparse-MLA fp8-KV decode path, the query tensor was being built by two back-to-back HBM-bound kernels: ConcatMLAQKernel and scaled_fp8_quant. Both are pure copy-with-scale, no compute. This PR routes forward_mqa through layer._decode_concat_quant_fp8_op, which torch.compile lowers to a single fused Triton kernel, eliminating the two kernel launches and improving throughput.

实现拆解

1. 在 forward_mqa 方法中前置判断 fp8_attention 状态，作为后续分支条件。
2. 当 q 为 tuple 且 fp8_attention 为 True 时，直接调用 layer._decode_concat_quant_fp8_op，该操作将 q_nope 和 q_pe 的拼接、转 fp32、缩放、钳位、转 fp8 合并在一个 Triton kernel 中完成，避免两个独立 HBM-bound kernel。
3. 非 fp8 路径保持原有的 concat_mla_q 机制（预分配缓冲区拷贝拼接）。
4. 在 fp8 量化块中，添加 q.dtype 检查，仅当 q 未量化时才执行 scaled_fp8_quant，避免融合后 key 重复量化。
5. 所有变更集中在 vllm/v1/attention/backends/mla/rocm_aiter_mla_sparse.py 的 forward_mqa 方法，共 +12/-7 行，无测试文件变更，但通过 GSM8K 端到端验证和性能 benchmark 保障正确性。

关键文件：

- vllm/v1/attention/backends/mla/rocm_aiter_mla_sparse.py（模块 注意力后端；类别 source；类型 core-logic；符号 forward_mqa）：唯一修改的文件，包含所有性能优化逻辑：融合 q 拼接与 fp8 量化、dtype 守卫、控制流调整。

关键符号：forward_mqa

关键源码片段

vllm/v1/attention/backends/mla/rocm_aiter_mla_sparse.py

唯一修改的文件，包含所有性能优化逻辑：融合 q 拼接与 fp8 量化、dtype 守卫、控制流调整。

```
def forward_mqa(
    self,
    q: torch.Tensor | tuple[torch.Tensor, torch.Tensor],
    kv_c_and_k_pe_cache: torch.Tensor,
    attn_metadata: ROCMAiterMLASparseMetadata,
    layer: AttentionLayer,
) -> tuple[torch.Tensor, torch.Tensor | None]:
    # NOTE(lucas): for sparse FlashMLA kernels use MQA 576/512 approach

    # 提前判断是否启用 fp8 量化，用于后续分支选择
    fp8_attention = self.kv_cache_dtype.startswith("fp8")

    # 处理 q 为 tuple（ql_nope, q_pe）的情况
    if isinstance(q, tuple):
        ql_nope, q_pe = q
        if fp8_attention:
            # 融合拼接与 fp8 量化：调用 MLACommonImpl 中已有 op，
            # 该 op 被 torch.compile 降低为一个 Triton kernel，避免两个独立 HBM-bound kernel
            q = layer._decode_concat_quant_fp8_op( # type: ignore[attr-defined]
                ql_nope, q_pe, layer._q_scale
            )
        else:
            # bf16-KV 路径：沿用原来的 concat buffer 拷贝拼接
            q = self.q_concat_buffer[: ql_nope.shape[0]]
            ops.concat_mla_q(ql_nope, q_pe, q)

    # 以下为原有逻辑（获取 topk 索引等）
    num_actual_toks = attn_metadata.num_actual_tokens
    assert self.topk_indices_buffer is not None
    topk_indices = self.topk_indices_buffer[:num_actual_toks]
    triton_convert_req_index_to_global_index(
        attn_metadata.req_id_per_token,
        attn_metadata.block_table,
        topk_indices,
        attn_metadata.paged_kv_indptr,
        attn_metadata.paged_kv_indices,
        BLOCK_SIZE=attn_metadata.block_size,
        NUM_TOPK_TOKENS=attn_metadata.topk_tokens,
    )

    # 写入 KV cache（含量化）
    if fp8_attention:
        kv_c_and_k_pe_cache = kv_c_and_k_pe_cache.view(current_platform.fp8_dtype())
        # 只有 q 还不是 fp8 时才需要显式量化，避免融合 kernel 已量化后的重复量化
        if q.dtype != current_platform.fp8_dtype():
            original_q_shape = q.shape
```

```
            q, _ = ops.scaled_fp8_quant(q.view(q.shape[0], -1), layer._q_scale)
            q = q.view(original_q_shape)
        # 后续 pad 和 attention 计算
        mla_padded_q = AiterMLAHelper.get_mla_padded_q(self.num_heads, q)
        attn_out = self._forward_mla(
            layer, mla_padded_q, kv_c_and_k_pe_cache, attn_metadata
        )
        return attn_out, None
```

评论区精华

- gemini-code-assist[bot] 建议使用 supports_quant_query_input 机制，让高层 MLAAttention 自动处理融合拼接量化，以保持后端设计一致。作者未直接回复，最终 PR 仍采用手动调用。
- dllehr-amd 指出一个多余的 # type: ignore[attr-defined] 注释，作者确认冗余并移除。
- dllehr-amd 要求测试更大并发（mc=128）以避免性能回归，作者补充了 mc=128 数据（TPOT 下降约 0.8%），验证无显著回归。
- rasmith 询问吞吐量数值，作者提供了 mc=4 和 mc=128 的 tok/s 和峰值 tok/s，证实了增益。
- 建议使用 supports_quant_query_input 机制替代手动调用 (design): 作者未直接回应，未采纳该建议，PR 保持手动调用方式。可考虑在后续清理 PR 中采纳。
- 冗余 type: ignore 注释 (style): 作者确认冗余并移除，已在后续 commit 中删除。
- 大并发性能验证请求 (testing): 作者提供 mc=128 的 TPOT 和吞吐量数据，显示仅有约 0.8% 下降，无明显回归。
- 吞吐量数据询问 (question): 作者提供了 mc=4 和 mc=128 的平均 tok/s 和峰值 tok/s，证明了性能增益。

风险与影响

- 风险：
  - 回归风险：bf16-KV 路径未修改，通过 dtype 守卫隔离；fp8 路径仅在层支持 _decode_concat_quant_fp8_op 时触发，该 op 已在 #42838 中测试，但若其行为变化可能影响其他调用点。
  - 性能风险：融合 kernel 依赖于 torch.compile 的降低能力，若 ROCm Triton 版本或 torch 版本变化，可能无法融合，但当前验证通过。
  - 测试覆盖：无直接单元测试，仅依赖 GSM8K 端到端正确性验证和性能基准测试，数值精度需持续监控。
  - 兼容性：改动局限在 ROCMAiterMLASparseImpl，不影响其他注意力后端。
- 影响：
  - 用户：ROCm 上使用 DeepSeek V3.2 且启用 fp8-KV 的推理场景获得约 3.7% TPOT 提升（mc=4），高并发下仍略有改善（约 0.8%）。
  - 系统：修改量极小，代码复杂度轻微增加，但执行效率提升。

- 团队：建立了复用 MLACommonImpl 中已有 op 的模式，未来类似后端可直接调用，期望后续统一使用 supports_quant_query_input 机制。
- 风险标记：有限测试覆盖（无单元测试），torch.compile 依赖，ROCm 特定路径

关联脉络

- PR #42838 [ROCm] Add *decode_concat_quant_fp8_op for MLA (inferred from context)*: 本 PR 复用该 PR 实现的 _decode_concat_quant_fp8_op 进行融合，并依赖其正确性；该 PR 已合并为本 PR 的依赖。