

PR #44977 完整报告

vllm-project/vllm

[ROCm][MLA] Fuse MLA q/kv RMSNorm + FP8 per-token quant in the FP8 attention path

合并时间: 2026-07-02 13:00

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/44977>

执行摘要

- 一句话: ROCm MLA FP8 路径融合 q/kv RMSNorm 与 per-token 量化
- 推荐动作: 建议精读该 PR, 特别是模式匹配 pass 的设计。它展示了如何处理编译 pass 间的流水线依赖——利用已有 pass 的规范形式作为匹配目标, 而非试图匹配更原始的图结构。此外, 关于 availability guard 取舍的讨论也值得关注。对于仅关心功能的读者, 可以直接合并该 PR 以获得 ROCm FP8 MLA 路径的性能提升。

功能与动机

在对 MLA 注意力进行 FP8 量化时, RocmAiterRMSNormQuantFusionPass 会将 q 侧的 rms_norm→fp8 per-token quant 折叠为 rocm_aiter_rmsnorm_fused_dynamic_quant, 导致 q 侧不再是裸 RMSNorm, 破坏 MLADualRMSNormFusionPass 的对称双 RMSNorm 模式匹配。这使得 FP8 路径下无法享受双 RMSNorm 融合的性能收益。该 PR 通过匹配这种不对称对 (q per-token quant + kv rms_norm), 将其降低为单个融合内核, 恢复 FP8 路径的融合能力。

实现拆解

该 PR 的实现分三个步骤:

1. 在 vllm/_aiter_ops.py 中注册新自定义算子: 添加 _fused_mla_dual_rms_norm_per_token_quant_impl (实际调用 AITER 的 fused_qk_rmsnorm_per_token_quant 内核)、_fused_mla_dual_rms_norm_per_token_quant_fake (fake 实现用于 trace) 以及通过 direct_register_custom_op 注册为 torch.ops.vllm.fused_mla_dual_rms_norm_per_token_quant。同时添加 get_fused_mla_dual_rms_norm_per_token_quant_op 访问器。初期包含 check_aiter_fused_qk_rmsnorm_per_token_quant 可用性检查, 在最终 commit 中随 AITER v0.1.16.post2 固定而被移除。
2. 在 vllm/compilation/passes/fusion/rocm_aiter_fusion.py 中新增 MLADualRMSPerTokenQuantPattern 类: 该类继承 VllmPatternReplacement, 定义 get_inputs 提供典型 MLA 形状的虚拟张量, pattern 属性描述 q 侧 rocm_aiter_rmsnorm_fused_dynamic_quant + kv 侧 vllm_ir.rms_norm 的图结构, replacement 属性利用新注册的 fused_mla_dual_rms_norm_per_token_quant op 实现融合替换。在 MLADualRMSNormFusionPass 的 __init__ 中, 当 AITER 提供该内核时, 为两个常用 epsilon 值 (1e-5、1e-6) 注册该 pattern。

3. 在 tests/compile/passes/test_fuse_mla_dual_rms_norm.py 中添加 MLADualRMSNormFp8PerTokenTestModel 和 test_fuse_mla_dual_rms_norm_fp8_per_token 测试：模型模拟 FP8 路径的 q norm+quant 和 kv norm，测试验证融合恰好触发一次且融合前后输出在 1e-2 误差内一致。
4. 在 docs/design/fusions.md 中更新文档，记录 FP8 attention 变体的融合过程。注意：该融合仅在 ROCm+AITER 启用时生效，默认在优化级别 O1 及以上自动开启。PR 最终版移除了动态可用性检查，依赖 AITER 固定版本提供内核。

关键文件：

- vllm/compilation/passes/fusion/rocm_aiter_fusion.py（模块 融合模式；类别 source；类型 core-logic；符号 MLADualRMSPerTokenQuantPattern, init, get_inputs, pattern）：定义了新的融合模式 MLADualRMSPerTokenQuantPattern，实现 FP8 路径下的双 RMSNorm+per-token 量化融合匹配与替换，是 PR 的核心逻辑所在。
- vllm/_aiter_ops.py（模块 AITER 接口；类别 source；类型 core-logic；符号 _fused_mla_dual_rms_norm_per_token_quant_impl, _fused_mla_dual_rms_norm_per_token_quant_fake, get_fused_mla_dual_rms_norm_per_token_quant_op）：注册新的 fused_mla_dual_rms_norm_per_token_quant 自定义算子，包含实际实现、fake 实现和访问器，是融合的内核调用入口。
- tests/compile/passes/test_fuse_mla_dual_rms_norm.py（模块 测试；类别 test；类型 test-coverage；符号 MLADualRMSNormFp8PerTokenTestModel, init, _dequant, forward）：新增 MLADualRMSNormFp8PerTokenTestModel 和对应测试用例，验证融合触发次数和数值一致性，确保 FP8 路径融合正确性。
- docs/design/fusions.md（模块 文档；类别 docs；类型 documentation）：更新文档以记录 FP8 attention 变体的融合信息，包括动机、代码位置和使用方式。

关键符号：_fused_mla_dual_rms_norm_per_token_quant_impl,
_fused_mla_dual_rms_norm_per_token_quant_fake,
get_fused_mla_dual_rms_norm_per_token_quant_op,
MLADualRMSPerTokenQuantPattern.init, MLADualRMSPerTokenQuantPattern.get_inputs,
MLADualRMSPerTokenQuantPattern.pattern,
MLADualRMSPerTokenQuantPattern.replacement

关键源码片段

vllm/compilation/passes/fusion/rocm_aiter_fusion.py

定义了新的融合模式 MLADualRMSPerTokenQuantPattern，实现 FP8 路径下的双 RMSNorm+per-token 量化融合匹配与替换，是 PR 的核心逻辑所在。

```
class MLADualRMSPerTokenQuantPattern(
    VllmPatternReplacement[
        ..., tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]
    ]
):
    # 引用先前注册的动态量化 op 和本模式要使用的 fused op
```

```
DYNAMIC_QUANT_OP = rocm_aiter_ops.get_rmsnorm_fused_dynamic_quant_op()
FUSED_OP = rocm_aiter_ops.get_fused_mla_dual_rms_norm_per_token_quant_op()
```

```
def __init__(self, epsilon: float) -> None:
    self._epsilon = epsilon
```

```
def get_inputs(self) -> list[torch.Tensor]:
    # 典型 MLA 几何尺寸: q_dim=256, kv_c_dim=128, k_pe_dim=64
    q_dim, kv_c_dim, k_pe_dim = 256, 128, 64
    return [
        self.empty_bf16(5, q_dim + kv_c_dim + k_pe_dim), # projected
        self.empty_bf16(q_dim), # q_weight
        self.empty_bf16(kv_c_dim), # kv_weight
    ]
```

```
@property
```

```
def pattern(self) -> Callable[..., tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]]:
```

```
    eps = self._epsilon
    dynamic_quant_op = self.DYNAMIC_QUANT_OP
```

```
def _pattern(projected, q_weight, kv_weight):
    # 拆分 q 和 kv 隐变量
    q_dim = q_weight.shape[0]
    kv_dim = projected.shape[-1] - q_dim
    kv_c_dim = kv_weight.shape[0]
    k_pe_dim = kv_dim - kv_c_dim
    q_c, kv_lora = projected.split([q_dim, kv_dim], dim=-1)
    kv_c, k_pe = kv_lora.split([kv_c_dim, k_pe_dim], dim=-1)
    # q 侧: RMSNorm + FP8 per-token 量化
    q_quant = dynamic_quant_op(x=q_c, weight=q_weight,
                               epsilon=eps, quant_dtype=FP8_DTYPE)
    # kv 侧: 仅 RMSNorm (bf16)
    kv_normed = vllm.ir.ops.rms_norm(kv_c, kv_weight, eps)
    return q_quant[0], q_quant[1], kv_normed, k_pe
```

```
    return _pattern
```

```
@property
```

```
def replacement(self) -> Callable[..., tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]]:
```

```
    eps = self._epsilon
    fused_op = self.FUSED_OP
```

```
def _replacement(projected, q_weight, kv_weight):
    # 与 pattern 相同的拆分逻辑
    q_dim = q_weight.shape[0]
    kv_dim = projected.shape[-1] - q_dim
    kv_c_dim = kv_weight.shape[0]
```

```

    k_pe_dim = kv_dim - kv_c_dim
    q_c, kv_lora = projected.split([q_dim, kv_dim], dim=-1)
    kv_c, k_pe = kv_lora.split([kv_c_dim, k_pe_dim], dim=-1)
    # 使用 fused op 一次计算 q_norm+quant 和 kv_norm
    at = fused_op(q_c, q_weight, kv_c, kv_weight, eps, eps)
    # 返回 (q_fp8, q_scale, kv_normed, k_pe)
    return at[0], at[1], at[2], k_pe

return _replacement

```

vllm/_aiter_ops.py

注册新的 fused_mla_dual_rms_norm_per_token_quant 自定义算子，包含实际实现、fake 实现和访问器，是融合的内核调用入口。

```

def _fused_mla_dual_rms_norm_per_token_quant_impl(
    q: torch.Tensor,
    q_weight: torch.Tensor,
    kv: torch.Tensor,
    kv_weight: torch.Tensor,
    q_epsilon: float,
    kv_epsilon: float,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    # 调用 AITER 的 fused_qk_rmsnorm_per_token_quant 内核
    from aiter.ops.fused_qk_rmsnorm_group_quant import fused_qk_rmsnorm_per_token_quant

    mq, nq = q.shape
    q_out = torch.empty((mq, nq), dtype=FP8_DTYPE, device=q.device)
    q_scale = torch.empty((mq, 1), dtype=torch.float32, device=q.device)
    kv_normed = torch.empty(kv.shape, dtype=kv.dtype, device=kv.device)

    fused_qk_rmsnorm_per_token_quant(
        q_out_quantized=q_out,
        q_out_scale=q_scale,
        q=q,
        q_weight=q_weight,
        q_epsilon=q_epsilon,
        k_out=kv_normed,
        k=kv,
        k_weight=kv_weight,
        k_epsilon=kv_epsilon,
        gemma_norm=False,
    )
    return q_out, q_scale, kv_normed

# 注册片段（位于 register_ops_once 内）
direct_register_custom_op(
    op_name="fused_mla_dual_rms_norm_per_token_quant",
    op_func=_fused_mla_dual_rms_norm_per_token_quant_impl,
    mutates_args=[],

```

```

    fake_impl=_fused_mla_dual_rms_norm_per_token_quant_fake,
)

@staticmethod
def get_fused_mla_dual_rms_norm_per_token_quant_op() -> OpOverload:
    # 外界通过此方法获取 op，用于 pattern replacement
    return torch.ops.vllm.fused_mla_dual_rms_norm_per_token_quant.default

```

评论区精华

评论中主要围绕三点：

1. Pass 顺序设计：Rohan138 建议将 MLADualRMSNormFusionPass 移至 RocmAiterRMSNormQuantFusionPass 之前，从而匹配原始的两个 RMSNorm+quant 模式。但 xaguilar-amd 解释当前顺序（norm+quant 先执行，再将结果与 kv norm 融合）更简单且安全，因为 q 侧被归一化为一个标准 op 后匹配，避免了重复匹配逻辑。该解释被接受。
 2. 代码洁癖：多位审查者要求减少 AI 生成的冗余注释。xaguilar-amd 在第二、三 commit 中集中精简了注释，获得认可。
 3. 可用性守卫：tjtanaa 指出 AITER 已固定到 v0.1.16.post2，应移除动态检查以简化代码并暴露潜在的 AITER 兼容性问题。xaguilar-amd 在最后 commit 中完全移除了 check_aiter_fused_qk_rmsnorm_per_token_quant 函数及相关守卫。
- Pass ordering: dual fusion after norm+quant (design): 当前顺序被保留，并在代码中添加了说明注释。
 - Verbose comments should be condensed (style): 注释被精简，Rohan138 随后 approve。
 - Remove availability guard after AITER pin (other): guard 被移除，依赖硬编码的 AITER 版本。

风险与影响

- 风险：风险包括：
 1. 依赖 AITER 特定内核版本：该融合依赖 AITER 的 fused_qk_rmsnorm_per_token_quant 内核，若未来 AITER 修改 API 或移除该内核，将导致编译 pass 失败或静默不触发。由于 PR 最终移除了动态检查，无法优雅降级。
 2. 数值精度：融合后的输出与分离计算的输出在测试中验证了 1e-2 误差，但不同模型或长度下可能存在更大偏差，特别是当 per-token scale 出现极端值时。
 3. 平台限定：该融合仅针对 ROCm，其他平台不受影响。但需确保在 CI 中覆盖 ROCm 测试。
 4. pass 生命周期：评论提到 fusion passes 可能很快被整体移除（#pr-merge-requests），若移除此 pass 则相关代码变为死代码，需及时清理。
 - 影响：影响范围：仅限使用 ROCm+AITER 且启用 FP8 MLA 注意力路径的模型（如 DeepSeek-V3、Kimi-K2）。性能影响：根据测试，在 Kimi-K2 MXFP4-AttnFP8 上吞吐提升约 1-2%，TPOT 降低 1-2%。无功能影响，融合结果与分离计算在容忍误差内一致。对团队：增加了少量维护负担，但代码已最小化且集中。无用户可见的配置变更，融合自动在 O1 及以上启用。-

风险标记: ROCm 限定, AITER 内核依赖, 精度敏感, pass 生命周期短

关联脉络

- 暂无明显关联 PR