PR #39242 完整报告

vllm-project/vllm

[ROCm] Add MLA dual RMS norm fusion (Q, KV) pass for DeepSeek/Kimi-K2

合并时间：2026-04-20 22:56

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39242

执行摘要

- 一句话：为 ROCm 平台添加 MLA 双 RMSNorm 融合优化，提升 DeepSeek-V3/Kimi-K2 模型性能。
- 推荐动作：建议工程团队精读此 PR，重点关注 MLADualRMSNormPattern 的模式设计如何动态推导 split 尺寸，以及 VllmFusionPatternMatcherPass 的使用范例。对于涉及图优化或硬件特定加速的开发者，此 PR 展示了如何通过 torch.inductor 模式匹配安全地融合复杂操作子图，具有较高参考价值。

功能与动机

根据 PR body 描述，未融合的 MLA 层需要运行两个独立的 RMSNorm 调用（分别处理 q 和 kv 压缩隐变量），导致每个层有 2 次内核启动。融合为一个内核可减少启动开销，提升模型推理性能，特别是在 DeepSeek-V3 和 Kimi-K2 这类具有 61 层 MLA 结构的模型上效果显著。关联 Issue #2442（ROCm/aiter）提供了底层的 fused_qk_rmsnorm HIP 内核支持。

实现拆解

1. 自定义操作注册：在 vllm/_aiter_ops.py 中新增 _fused_mla_dual_rms_norm_impl 和 _fused_mla_dual_rms_norm_fake 函数，并通过 direct_register_custom_op 注册为 fused_mla_dual_rms_norm 操作，封装 AITer 的 fused_qk_rmsnorm 内核。
2. 模式匹配 pass 实现：在 vllm/compilation/passes/fusion/rocm_aiter_fusion.py 中新增 MLADualRMSNormPattern 类（继承 VllmPatternReplacement），定义模式识别和替换逻辑，将连接的子图（包含 split 操作和两个 rms_norm 调用）重写为单个融合操作；MLADualRMSNormFusionPass 类（继承 VllmFusionPatternMatcherPass）负责在 FX 图中应用该模式。
3. 配置系统集成：在 vllm/config/vllm.py 中添加 enable_mla_dual_rms_norm_fusion 函数，根据 AITer 可用性控制融合开关；在 vllm/config/compilation.py 的 PassConfig 中新增 fuse_mla_dual_rms_norm 布尔字段，并添加 ROCm 平台检查；在 vllm/compilation/passes/pass_manager.py 中注册该 pass 到 pass 流水线。
4. 测试配套：新增 tests/compile/passes/test_fuse_mla_dual_rms_norm.py 单元测试，包含 MLADualRMSNormTestModel 模型和 test_fuse_mla_dual_rms_norm 测试函数，验证模式匹配、操作替换和数值正确性。

5. 文档更新：更新 docs/design/fusions.md 和 docs/design/optimization_levels.md，记录该融合 pass 的使用说明和配置项。

关键文件：

- vllm/compilation/passes/fusion/rocm_aiter_fusion.py（模块 编译融合；类别 source；类型 core-logic；符号 MLADualRMSNormPattern, init, get_inputs, pattern）：核心实现文件，包含 MLA 双 RMSNorm 融合的模式匹配和 pass 逻辑，定义了如何识别和重写 FX 图。
- tests/compile/passes/test_fuse_mla_dual_rms_norm.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 MLADualRMSNormTestModel, init, forward, ops_in_model_before）：单元测试文件，验证融合 pass 的正确性，包括模式匹配、操作替换和数值精度。
- vllm/_aiter_ops.py（模块 操作注册；类别 source；类型 core-logic；符号 _fused_mla_dual_rms_norm_impl, _fused_mla_dual_rms_norm_fake, get_fused_mla_dual_rms_norm_op）：注册 fused_mla_dual_rms_norm 自定义操作，作为 AITer 内核的包装器。
- vllm/config/vllm.py（模块 配置系统；类别 source；类型 configuration；符号 enable_mla_dual_rms_norm_fusion）：添加融合使能函数和优化级别配置，控制该 pass 的触发条件。
- vllm/config/compilation.py（模块 配置系统；类别 source；类型 configuration）：在 PassConfig 中添加 fuse_mla_dual_rms_norm 字段，并提供平台检查逻辑。

关键符号：MLADualRMSNormPattern, MLADualRMSNormFusionPass, _fused_mla_dual_rms_norm_impl, enable_mla_dual_rms_norm_fusion, test_fuse_mla_dual_rms_norm

关键源码片段

vllm/compilation/passes/fusion/rocm_aiter_fusion.py

核心实现文件，包含 MLA 双 RMSNorm 融合的模式匹配和 pass 逻辑，定义了如何识别和重写 FX 图。

```
class MLADualRMSNormPattern(
    VllmPatternReplacement[..., tuple[torch.Tensor, torch.Tensor, torch.Tensor]]
):
    """
    融合MLA注意力中配对的q_a_layernorm和kv_a_layernorm到AITER的fused_qk_rmsnorm
    HIP内核。
    目标FX图模式（未融合，vllm_ir阶段）：
        gemm -> split_with_sizes([q_dim, kv_dim])
            +-- q_c -> vllm_ir.rms_norm(q_c, q_w, eps)
            +-- kv_lora -> split_with_sizes([kv_c_dim, k_pe_dim])
                    +-- kv_c -> vllm_ir.rms_norm(kv_c, kv_w, eps)
                    +-- k_pe
    """

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

```
        self._epsilon = epsilon # 设置 epsilon 参数，用于模式匹配中的 RMSNorm 计算

    def get_inputs(self) -> list[torch.Tensor]:
        # 提供虚拟输入用于模式验证，尺寸任意但保持维度一致性
        q_dim, kv_c_dim, k_pe_dim = 8, 4, 2
        return [
            self.empty_bf16(5, q_dim + kv_c_dim + k_pe_dim), # projected 输入
            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]]:
        eps = self._epsilon
        def _pattern(projected: torch.Tensor, q_weight: torch.Tensor, kv_weight: torch.Tensor):
            q_dim = q_weight.shape[0] # 动态获取 q 维度
            kv_dim = projected.shape[-1] - q_dim # 计算 kv 总维度
            kv_c_dim = kv_weight.shape[0] # 动态获取 kv_c 维度
            k_pe_dim = kv_dim - kv_c_dim # 计算 k_pe 维度
            q_c, kv_lora = projected.split([q_dim, kv_dim], dim=-1) # 第一次 split
            kv_c, k_pe = kv_lora.split([kv_c_dim, k_pe_dim], dim=-1) # 第二次 split
            q_normed = vllm.ir.ops.rms_norm(q_c, q_weight, eps) # q 的 RMSNorm
            kv_normed = vllm.ir.ops.rms_norm(kv_c, kv_weight, eps) # kv 的 RMSNorm
            return q_normed, kv_normed, k_pe # 返回三个输出
        return _pattern

    @property
    def replacement(self) -> Callable[..., tuple[torch.Tensor, torch.Tensor, torch.Tensor]]:
        eps = self._epsilon
        def _replacement(projected: torch.Tensor, q_weight: torch.Tensor, kv_weight: torch.
        Tensor):
            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)
            # 使用融合操作替换两个独立的 RMSNorm 调用
            q_normed, kv_normed = torch.ops.vllm.fused_mla_dual_rms_norm(
                q_c, q_weight, kv_c, kv_weight, eps, eps
            )
            return q_normed, kv_normed, k_pe
        return _replacement
```

tests/compile/passes/test_fuse_mla_dual_rms_norm.py

单元测试文件，验证融合 pass 的正确性，包括模式匹配、操作替换和数值精度。

```
class MLADualRMSNormTestModel(torch.nn.Module):
    """
```

```
    最小化模型，复现MLA双RMSNorm模式：
        linear -> split([q_dim, kv_dim])
            +-- q_c -> rms_norm(q_w, eps) -> linear
            +-- kv_lora -> split([kv_c_dim, k_pe_dim])
                    +-- kv_c -> rms_norm(kv_w, eps)
                    +-- k_pe
    """
    def __init__(self, hidden_size: int, q_dim: int = 1536, kv_c_dim: int = 512, k_pe_dim: int = 64,
    eps: float = 1e-6):
        super().__init__()
        self.q_dim = q_dim
        self.kv_dim = kv_c_dim + k_pe_dim
        self.kv_c_dim = kv_c_dim
        self.k_pe_dim = k_pe_dim
        self.proj = torch.nn.Linear(hidden_size, q_dim + self.kv_dim, bias=False) # 投影层
        self.q_norm = RMSNorm(q_dim, eps=eps) # q 的 RMSNorm 层
        self.kv_norm = RMSNorm(kv_c_dim, eps=eps) # kv 的 RMSNorm 层
        self.q_b_proj = torch.nn.Linear(q_dim, hidden_size, bias=False) # 后续线性层

    def forward(self, x: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
        x = torch.relu(x) # 避免输入直接作为模式节点
        projected = self.proj(x) # 投影操作
        q_c, kv_lora = projected.split([self.q_dim, self.kv_dim], dim=-1) # 第一次 split
        kv_c, k_pe = kv_lora.split([self.kv_c_dim, self.k_pe_dim], dim=-1) # 第二次 split
        q_normed = self.q_norm(q_c) # 原始 q RMSNorm
        kv_normed = self.kv_norm(kv_c) # 原始 kv RMSNorm
        q_out = self.q_b_proj(q_normed) # 后续处理
        return q_out, kv_normed, k_pe # 返回三个输出，用于数值比较

    def ops_in_model_before(self):
        return [torch.ops.vllm_ir.rms_norm.default] # 融合前期望的操作

    def ops_in_model_after(self):
        return [torch.ops.vllm.fused_mla_dual_rms_norm.default] # 融合后期望的操作
```

评论区精华

核心讨论点：

- 图拓扑顺序风险：gemini-code-assist[bot] 指出初始实现中手动遍历节点可能导致输入节点未正确提升，引发编译失败。开发者 rbrugaro-amd 随后重构为模式匹配方案以规避此问题。
- 代码结构与最佳实践：Rohan138 建议使用 torch.inductor 的 PatternMatcher 而非手动迭代，并将 pass 整合到现有 rocm_aiter_fusion.py 文件中；ProExpertProg 进一步推荐使用 VllmFusionPatternMatcherPass 基类使代码更简洁，这些建议均被采纳。
- 配置条件细化：Rohan138 对 enable_mla_dual_rms_norm_fusion 函数仅检查 AITer 可用性提出疑问，但 ProExpertProg 认为当前条件足够，未引入额外限制。
- 文档与命名规范：ProExpertProg 请求在文档中添加关于 Inductor 默认融合的说明；Rohan138 指出配置日志中的“AITer”拼写应统一为“AITER”。

- 图拓扑顺序与模式匹配重构 (design): 开发者重构代码，采用 PatternMatcher 和 VllmFusionPatternMatcherPass，解决了拓扑风险并遵循最佳实践。
- 配置条件与文档更新 (design): 配置逻辑保持原样，文档已更新以记录融合 pass 的详细信息。
- 代码风格与命名规范 (style): 开发者采纳格式建议并修正拼写，确保代码一致性。

风险与影响

- 风险：
  1. 平台依赖性风险：融合仅适用于 ROCm 平台且依赖外部 AITer 库（PR #2442），若 AITer 不可用或版本不兼容，融合将自动禁用，但可能引发用户困惑。
  2. 模式匹配健壮性风险：MLADualRMSNormPattern 依赖于特定的 FX 图结构（如 split 尺寸和 rms_norm 调用顺序），若模型图结构变化（例如不同 MLA 变体），可能导致匹配失败或误匹配。
  3. 数值精度风险：融合内核 fused_qk_rmsnorm 与原始两个 RMSNorm 操作的数值等价性依赖 AITer 实现，虽经单元测试验证，但在边缘情况（如极端 epsilon 值）下仍需监控。
  4. 编译时性能风险：模式匹配增加了图遍历开销，可能轻微影响编译时间，但鉴于融合仅在优化等级≥O1 时触发，影响可控。
- 影响：
  1. 用户影响：对于使用 DeepSeek-V3 或 Kimi-K2 模型的 ROCm 用户，在启用优化（默认 O1 及以上）后可获得约 1.02 倍的吞吐量提升（基于 PR 中性能数据），无需修改模型代码或配置。
  2. 系统影响：减少内核启动次数，降低 GPU 调度开销，有助于提高硬件利用率；但仅影响 MLA 注意力层，不改变其他模型组件或 API 接口。
  3. 团队影响：引入了新的编译 pass 和配置项，增加了 ROCm 专用优化模块的复杂性，需团队在后续维护中熟悉模式匹配框架和 AITer 集成。 - 风险标记：依赖外部 AITer, ROCm 平台特定，模式匹配复杂性，数值精度验证

关联脉络

- PR #37712 Properly enable wvSplitK fp8 path for RDNA: 同为 ROCm 平台优化，涉及量化路径和内核启用，显示 ROCm 持续性能改进趋势。
- PR #40152 mxfp8 online quant move to new frontend: 涉及量化重构，与本 PR 的融合优化同属编译和内核优化范畴，共享类似的技术模式。
- PR #38093 [Bugfix] Fix scaled_mm output narrowing for 3D input tensors: 修复 ROCm 相关内核问题，凸显平台特定优化中兼容性和正确性的重要性。