PR #40710 完整报告

vllm-project/vllm

[Aiter][ROCm] RMSNormGated+GroupedQuantFP8 fusion

合并时间：2026-05-15 03:37

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40710

执行摘要

- 一句话：ROCm 上融合 RMSNormGated 与 FP8 分组量化提升性能
- 推荐动作：值得阅读该 PR 的实现，尤其是 torch.fx 级别的模式匹配集成方式、与 AITER 的协作模式以及 kernel 可用性检查的优雅回退。对计划支持类似融合优化的开发者有参考价值。设计决策中的折衷（如 head_dim 的硬编码、match_aiter_quant 的处理）和后续迁移到 vLLM IR 的规划也值得关注。

功能与动机

在支持 GatedDeltaNetAttention 的模型（如 Qwen3-Next-80B-A3B-Instruct-FP8）中，每个注意力头的输出经过 RMSNormGated、reshape 到完整隐维度、再进行分组 FP8 量化后送入输出投影。分解后的序列在 torch.compile 下生成多个小 kernel，导致多次 GPU kernel 启动和中间内存读写。融合为一个 Triton kernel 后可消除这些开销，在低并发场景下带来 1-3% 的性能提升。参考 PR Body："A 9us set of 2 kernels can be combined to 4.5us"。

实现拆解

1. 注册 AITER 融合算子：在 vllm/_aiter_ops.py 中新增 _rocm_aiter_fused_rms_gated_fp8_group_quant_impl 和对应的 fake 实现，并通过 register_ops_once() 注册为自定义 op (rocm_aiter_fused_rms_gated_fp8_group_quant)。同时扩展 are_gdn_triton_kernels_available() 检查该 kernel 是否可用，确保向后兼容。
2. 提取静态计算方法：将 RMSNormGated.forward_native 重构为 forward_static 静态方法（vllm/model_executor/layers/layernorm.py），使模式匹配器和原生计算共享同一份 PyTorch 实现，避免重复并保证数值一致性。
3. 实现模式匹配器：在 matcher_utils.py 中新增 MatcherRMSNormGated 类，继承 MatcherCustomOp。它提供 forward_native（调用 RMSNormGated.forward_static）和 forward_custom（调用 AITER 的 rmsnorm 函数），使模式可同时跟踪自定义和原生路径。
4. 实现融合模式并注册：在 rocm_aiter_fusion.py 中新增 AiterRMSNormGatedFp8GroupQuantPattern，匹配 "RMSNormGated → reshape → Group FP8 Quant" 的图子序列并替换为单个融合 kernel。注册前通过 fold_consecutive_reshapes（新增于 vllm_inductor_pass.py）折叠连续 reshape，确保 pattern 匹配成功。从 get_layers_from_vllm_config 动态推断 num_heads 和 head_dim。模式注册在 RocmAiterRMSNormQuantFusionPass 中，并由 are_gdn_triton_kernels_available() 门控。

5. 添加测试：在 tests/compile/passes/test_fusion.py 中新增 TestGatedModel 模拟 GDN 注意力输出路径，覆盖 aiter 量化、非 aiter 量化、per-token 动态量化等正例，以及错误 group_shape、per-tensor 量化等反例。

关键文件：

- vllm/compilation/passes/fusion/rocm_aiter_fusion.py（模块 融合优化；类别 source；类型 core-logic；符号 AiterRMSNormGatedFp8GroupQuantPattern, init, register, pattern）：核心融合 pass，新增 AiterRMSNormGatedFp8GroupQuantPattern 模式匹配与替换逻辑，是 PR 的主实现文件
- vllm/compilation/passes/fusion/matcher_utils.py（模块 模式匹配；类别 source；类型 core-logic；符号 MatcherRMSNormGated, init, inputs, forward_custom）：新增 MatcherRMSNormGated 类，为模式匹配提供统一的自定义 / 原生路径追踪
- vllm/model_executor/layers/layernorm.py（模块 模型层；类别 source；类型 data-contract；符号 forward_native, forward_static）：重构 RMSNormGated.forward_native 为 forward_static 静态方法，供 matcher 共享，减少重复
- vllm/_aiter_ops.py（模块 后端接口；类别 source；类型 core-logic；符号 _rocm_aiter_fused_rms_gated_fp8_group_quant_impl, _rocm_aiter_fused_rms_gated_fp8_group_quant_fake, get_fused_rms_gated_fp8_group_quant_op）：注册自定义融合算子 (rocm_aiter_fused_rms_gated_fp8_group_quant) 及 fake 实现，扩展 kernel 可用性检查
- vllm/compilation/passes/vllm_inductor_pass.py（模块 编译通道；类别 source；类型 core-logic；符号 fold_consecutive_reshapes）：新增 fold_consecutive_reshapes 预处理 pass，确保 pattern 能匹配编译图
- tests/compile/passes/test_fusion.py（模块 测试；类别 test；类型 test-coverage；符号 TestGatedModel, init, forward, ops_in_model_after）：新增 TestGatedModel 及多项测试用例，覆盖正例和反例，保障融合正确性

关键符号：AiterRMSNormGatedFp8GroupQuantPattern.init, AiterRMSNormGatedFp8GroupQuantPattern.register, MatcherRMSNormGated.init, MatcherRMSNormGated.forward_custom, MatcherRMSNormGated.forward_native, RMSNormGated.forward_static, _rocm_aiter_fused_rms_gated_fp8_group_quant_impl, _rocm_aiter_fused_rms_gated_fp8_group_quant_fake, fold_consecutive_reshapes, are_gdn_triton_kernels_available

关键源码片段

vllm/compilation/passes/fusion/rocm_aiter_fusion.py

核心融合 pass，新增 AiterRMSNormGatedFp8GroupQuantPattern 模式匹配与替换逻辑，是 PR 的主实现文件

```
class AiterRMSNormGatedFp8GroupQuantPattern(AiterRMSNormQuantPattern):
    """
    Matches decomposed RMSNormGated + reshape + group FP8 quant and replaces
```

```
with rocm_aiter_fused_rms_gated_fp8_group_quant.
The norm operates per-head on (N*H, D) tensors. The compiler folds the
reshape chain so after norm the result goes through reshape->merge->quant.
"""
FUSED_OP = rocm_aiter_ops.get_fused_rms_gated_fp8_group_quant_op()

def __init__(
    self,
    epsilon: float,
    quant_dtype: torch.dtype,
    group_shape: GroupShape,
    num_heads: int,
    head_dim: int,
    match_aiter_quant: bool = True,
    symmetric: bool = True,
) -> None:
    # 构造 quant key, 使用 group_shape(1,128) 匹配 FP8 分组量化
    scale = ScaleDesc(torch.float32, False, group_shape)
    key = FusedRMSQuantKey(
        fused_add=False,
        quant=QuantKey(dtype=quant_dtype, scale=scale, symmetric=symmetric),
    )
    super().__init__(epsilon, key, match_aiter_quant)
    # 创建 RMSNormGated 模式匹配器
    self.rmsnorm_gated_matcher = MatcherRMSNormGated(epsilon)
    self.num_heads = num_heads
    self.head_dim = head_dim

def register(self, pm_pass: PatternMatcherPass) -> None:
    num_heads = self.num_heads
    head_dim = self.head_dim
    hidden_dim = num_heads * head_dim
    quant_matcher = self.quant_matcher

    # 定义匹配模式：输入 x,z,weight -> RMSNormGated -> reshape -> 量化
    def pattern(
        x: torch.Tensor,
        z: torch.Tensor,
        weight: torch.Tensor,
    ) -> tuple[torch.Tensor, torch.Tensor]:
        normed = self.rmsnorm_gated_matcher(x, z, weight)
        merged = normed.reshape(-1, hidden_dim)
        quant_out, scales_out = quant_matcher(merged)
        return quant_out, scales_out

    # 定义替换：单一融合 kernel，输出 reshape 匹配原始量化输出形状
    def replacement(
        x: torch.Tensor,
        z: torch.Tensor,
```

```
        weight: torch.Tensor,
    ) -> tuple[torch.Tensor, torch.Tensor]:
        fused = self.FUSED_OP(
            x=x, weight=weight, bias=None, z=z,
            eps=self.epsilon, norm_before_gate=True,
            activation="silu", group_size=head_dim,
        )
        return fused[0].reshape(-1, hidden_dim), fused[1].reshape(-1, num_heads)

    # 构建示例输入 (2 个 token)
    n_tokens = 2
    example_args = [
        self.empty(n_tokens * num_heads, head_dim), # x
        self.empty(n_tokens * num_heads, head_dim), # z
        self.empty(head_dim), # weight
    ]

    def trace_fn(*args):
        gm = pm.fwd_only(*args)
        view_to_reshape(gm)
        fold_consecutive_reshapes(gm) # 折叠连续 reshape 以匹配编译图
        return gm

    pm.register_replacement(pattern, replacement, example_args, trace_fn, pm_pass)
```

评论区精华

- 全局 monkey-patching 安全性：gemini-code-assist[bot] 指出 pm.fx_to_pattern 忽略所有 int 和 SymInt 类型极其危险，可能导致错误匹配不同 group size。作者 tpopp 回应暂未找到更好方案，随后在后续 commit 中完全移除了该 monkey-patch，改用 fold_consecutive_reshapes 并确保 pattern 使用正确符号类型。问题已解决。
- 模式注册缺少 head_dim 校验及 match_aiter_quant 遍历：bot 发现新模式未遍历 match_aiter_quant 值（只匹配 aiter 量化路径）且未检查 head_dim==128，可能漏匹配或导致数值错误。作者添加了 head_dim 检查，并说明在分组量化场景下 match_aiter_quant 因 aiter 特殊路径不能简单通用。审查后认可。
- 未来迁移到 vLLM IR：ProExpertProg 建议将 MatcherRMSNormGated 后续迁移到 vLLM IR (#38798)，tpopp 表示同意并计划跟进。
  - 全局 monkey-patch fx_to_pattern 的安全风险 (correctness): 作者在后续 commit 中移除了该 monkey-patch，改由 fold_consecutive_reshapes 和保留正确符号类型解决。
  - 模式注册缺少 head_dim 校验及 match_aiter_quant 遍历 (correctness): 作者添加了 head_dim 检查，并说明在分组量化场景下 match_aiter_quant 因 aiter 特殊路径无法通用。审查者认可。
  - 未来将 MatcherRMSNormGated 迁移至 vLLM IR (design): tpopp 同意作为 follow-up，当前保持基于 MatcherCustomOp 的实现。

风险与影响

- 风险：
  1. 全局 monkey-patch 已移除：初版使用 pm.fx_to_pattern 忽略 int/SymInt，但已被删除，风险解除。
  2. group_size 硬编码 128：融合 kernel 仅支持 GroupShape(1,128)，模式注册时添加了 head_dim==128 检查，确保匹配安全。若未来模型使用不同 group size，需扩展模式。
  3. ROCm 专属：该融合 pass 仅在 ROCm + AITER 环境下生效，其他平台无影响。
  4. 精度可验证：在 gsm8k 5-shot 上基准与融合版本准确率无统计显著差异（±0.9pp 内），数值安全。
  5. 回退条件明确：如果 AITER 版本缺少 GDN kernel，are_gdn_triton_kernels_available() 返回 False，pass 自动跳过，功能不受影响。 - 影响：影响范围：仅针对 ROCm 平台上使用 GatedDeltaNetAttention 的模型（如 Qwen3-Next-80B），其他模型或平台无影响。若 AITER 缺少必需 kernel，pass 自动回退，功能不受影响。性能收益：在 AMD MI355x 上，ISL/OSL=1024，concurrency=4 时输出吞吐 +2.3%，TPOT 延迟 -2.5%，且加速比随并发降低更明显。维护成本：引入新的 Pattern 和 Matcher 类，代码集中在编译优化层，不影响模型加载或核心推理路径。测试覆盖了正反例，保障后续修改安全。
- 风险标记：全局 monkey-patch 已移除，group_size 硬编码 128，仅在 ROCm 上生效，精度无统计差异

关联脉络

- PR #2423 [Triton] optimized decode kernels for Qwen3-Next model: 此 PR 融合 kernel 的上游依赖，提供 fused_rms_gated_fp8_group_quant 的 Triton 实现
- PR #38798 vLLM IR compilation pass: 讨论中提议将 MatcherRMSNormGated 未来迁移至此 IR 以简化维护