PR #41825 完整报告

vllm-project/vllm

[ROCm][Perf] Fix RMSNorm+Quant fusion for gfx950 (non-fnuz)

合并时间：2026-05-12 03:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41825

执行摘要

- 一句话：修复 gfx950 上 RMSNorm+FP8 融合，延迟降 3.3%
- 推荐动作：值得精读该 PR，尤其是 matcher_utils.py 的修正和 DoubleAiterRMSFp8GroupQuantPattern 的声明式模式实现。它展示了从手动 FX 图变换到声明式模式匹配的演进思路，以及 view-tolerant 变体处理实际生产图中常见噪声的经验。设计决策（重复 rms_norm 而非保留未融合的 16 位读取）也有借鉴意义。建议在撰写自定义编译 pass 时参考此模式。

功能与动机

PR 描述指出，gfx950(non-fnuz) 上 AITER RMSNorm+GroupedQuantFP8 融合内核被静默跳过，导致性能未达预期。两个根因：一是 MatcherQuantFP8 在非 fnuz 平台错误选择了 triton_per_token_group_quant_fp8 而非 rocm_aiter_group_fp8_quant；二是 DSv3.2 的 FX 图中 RMSNorm 输出被多个量化操作共享，违反 1-to-1 模式匹配约束。

实现拆解

1. 修正算子选择（matcher_utils.py）：移除 MatcherQuantFP8 中 is_fp8_fnuz() 条件判断，当 match_rocm_aiter=True 时始终使用 rocm_aiter_ops.get_group_quant_op()，确保 gfx950 匹配正确算子。
2. 新增 DoubleQuant 模式（rocm_aiter_fusion.py）：定义 DoubleAiterRMSFp8GroupQuantPattern 类，匹配一个 rms_norm 输出被两个不同 group_fp8_quant 消费的 fan-out 图，替换为两个独立融合的 rms_norm_group_fp8_quant 操作。
3. 添加 view-tolerant 变体（rocm_aiter_fusion.py）：DoubleAiterRMSFp8GroupQuantViewPattern 匹配中间有 view/reshape 的 fan-out 图（DSv3.2 MLA indexer 特有形状），通过 torch._inductor.fx_passes.post_grad.view_to_reshape 将 view 转化为 reshape，再匹配模式。
4. 调整 UUID 缓存键（pass_manager.py）：恢复 clone_elimination 在非 gfx950 上的 UUID 参与缓存键，仅在 gfx950 上 pop 避免缓存未命中。
5. 添加单元测试（tests/compile/passes/test_double_aiter_rms_quant_fusion.py）：用 _NoViewDoubleQuantModel 和 _ViewDoubleQuantModel 两个模型分别测试无 view 和有 view 场景，参数化运行 RocmAiterRMSNormQuantFusionPass 并验证融合后的图中出现

rocm_aiter_rmsnorm_fp8_group_quant 节点。

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 DoubleAiterRMSFp8GroupQuantPattern, init, register, pattern）：核心变更文件，新增 DoubleAiterRMSFp8GroupQuantPattern 和 DoubleAiterRMSFp8GroupQuantViewPattern 两个模式类，分别处理无 view 和有 view 的 1-to-2 fan-out 融合场景。同时优化了 pass 的 uuid 生成。是整个 PR 的核心实现。
- vllm/compilation/passes/fusion/matcher_utils.py（模块 编译匹配；类别 source；类型 core-logic）：关键的算子选择修复：移除 is_fp8_fnuz() 条件，使 gfx950 上始终选择正确的 rocm_aiter_group_fp8_quant，是融合生效的基础。
- tests/compile/passes/test_double_aiter_rms_quant_fusion.py（模块 编译测试；类别 test；类型 test-coverage；符号 _NoViewDoubleQuantModel, init, forward, _ViewDoubleQuantModel）：新增单元测试，覆盖无 view 和有 view 两种 fan-out 形状，通过参数化模型验证 DoubleQuant 模式正确触发融合，提供回归保护。

关键符号：DoubleAiterRMSFp8GroupQuantPattern, DoubleAiterRMSFp8GroupQuantPattern.register, DoubleAiterRMSFp8GroupQuantViewPattern, DoubleAiterRMSFp8GroupQuantViewPattern.trace_with_view_to_reshape, MatcherQuantFP8.init, test_double_aiter_rms_fp8_group_quant_fusion

关键源码片段

vllm/compilation/passes/fusion/rocm_aiter_fusion.py

核心变更文件，新增 DoubleAiterRMSFp8GroupQuantPattern 和 DoubleAiterRMSFp8GroupQuantViewPattern 两个模式类，分别处理无 view 和有 view 的 1-to-2 fan-out 融合场景。同时优化了 pass 的 uuid 生成。是整个 PR 的核心实现。

```
class DoubleAiterRMSFp8GroupQuantPattern(AiterRMSNormQuantPattern):
    """
    Pattern matching ``rms_norm`` whose output feeds *two* distinct
    ``rocm_aiter_group_fp8_quant`` consumers, replacing it with two
    independent fused ``rms_norm_group_fp8_quant`` ops.

    Repeating the rms_norm in the replacement is preferable to leaving
    the fused 16-bit rms output materialized for two unfused quant
    consumers, and matches what the previous manual graph surgery
    achieved by cloning the rms_norm node.
    """

    FUSED_OP = rocm_aiter_ops.get_rmsnorm_group_fused_quant_op()

    def __init__(
        self,
        epsilon: float,
        quant_dtype: torch.dtype,
```

```
        group_shape: GroupShape,
        match_aiter_quant: bool = True,
        symmetric: bool = True,
    ) -> None:
        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)

    def register(self, pm_pass: PatternMatcherPass) -> None:
        # 定义 pattern：一个 rms_norm 输出连接到两个相同的量化操作
        def pattern(
            input: torch.Tensor,
            weight: torch.Tensor,
        ) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
            result_rms = torch.ops.vllm_ir.rms_norm(input, weight, self.epsilon)
            result1, scale1 = self.quant_matcher(result_rms)
            result2, scale2 = self.quant_matcher(result_rms)
            return result1, scale1, result2, scale2

        # 定义 replacement：用两个独立的 fused op 替代
        def replacement(
            input: torch.Tensor,
            weight: torch.Tensor,
        ) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
            at1 = self.FUSED_OP(
                x=input, weight=weight, variance_epsilon=self.epsilon, group_size=128,
            )
            at2 = self.FUSED_OP(
                x=input, weight=weight, variance_epsilon=self.epsilon, group_size=128,
            )
            return at1[0], at1[1], at2[0], at2[1]

        pm.register_replacement(
            pattern, replacement,
            [self.empty(5, 16), self.empty(16)], # 示例输入
            pm.fwd_only, pm_pass,
        )

class DoubleAiterRMSFp8GroupQuantViewPattern(AiterRMSNormQuantPattern):
    """
    View-tolerant variant that matches the same fan-out but with a
    ``view``/``reshape`` between the ``rms_norm`` output and the two
    ``rocm_aiter_group_fp8_quant`` consumers.

    This shape arises in DeepSeek-V3.2's MLA indexer q_c norm, where
    ``Fp8BlockScaledMMLinearKernel.apply_weights`` inserts a 2D-flatten
```

```
    view before each quant op.
    """
    ...
    @staticmethod
    def trace_with_view_to_reshape(graph: fx.Graph) -> None:
        # 将图中的 view 节点转换为 reshape，使 pattern 能匹配
        view_to_reshape(graph, skip_constructors=True)
        # 这里还可以处理连续 reshape 的折叠
```

tests/compile/passes/test_double_aiter_rms_quant_fusion.py

新增单元测试，覆盖无 view 和有 view 两种 fan-out 形状，通过参数化模型验证 DoubleQuant 模式正确触发融合，提供回归保护。

```
class _NoViewDoubleQuantModel(torch.nn.Module):
    """``rms_norm -> 2x group_fp8_quant`` fan-out (Kimi-K2.5 / DSR1 shape)."""
    def __init__(self) -> None:
        super().__init__()
        self.weight = torch.nn.Parameter(torch.ones(HIDDEN_SIZE, dtype=torch.bfloat16))

    def forward(self, x: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.
    Tensor]:
        x = torch.relu(x)
        rms = torch.ops.vllm_ir.rms_norm(x, self.weight, EPS)
        q1, s1 = torch.ops.vllm.rocm_aiter_group_fp8_quant.default(rms, GROUP_SIZE)
        q2, s2 = torch.ops.vllm.rocm_aiter_group_fp8_quant.default(rms, GROUP_SIZE)
        return q1, s1, q2, s2


class _ViewDoubleQuantModel(torch.nn.Module):
    """``rms_norm -> view -> 2x group_fp8_quant`` fan-out (DSv3.2 shape)."""
    def __init__(self) -> None:
        super().__init__()
        self.weight = torch.nn.Parameter(torch.ones(HIDDEN_SIZE, dtype=torch.bfloat16))

    def forward(self, x: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.
    Tensor]:
        x = torch.relu(x)
        rms = torch.ops.vllm_ir.rms_norm(x, self.weight, EPS)
        view = rms.view(-1, rms.shape[-1])
        q1, s1 = torch.ops.vllm.rocm_aiter_group_fp8_quant.default(view, GROUP_SIZE)
        q2, s2 = torch.ops.vllm.rocm_aiter_group_fp8_quant.default(view, GROUP_SIZE)
        return q1, s1, q2, s2


@pytest.mark.parametrize("model_cls", [_NoViewDoubleQuantModel, _ViewDoubleQuantModel],
ids=["no_view", "with_view"])
@pytest.mark.skipif(not is_aiter_found_and_supported(), reason="Only test on ROCm with AITER
installed and supported")
def test_double_aiter_rms_fp8_group_quant_fusion(model_cls: type[torch.nn.Module],
```

```
monkeypatch: pytest.MonkeyPatch) -> None:
    """
    Both fan-out shapes must fuse into ``rocm_aiter_rmsnorm_fp8_group_quant``.
    Failure on the ``with_view`` parametrization is a regression on the
    DSv3.2 q_c norm path that this PR's view-tolerant pattern is intended to cover.
    """
    torch._dynamo.reset()
    vllm_config = VllmConfig(
        model_config=ModelConfig(dtype=torch.bfloat16),
        compilation_config=CompilationConfig(
            mode=CompilationMode.VLLM_COMPILE,
            custom_ops=["+rms_norm", "+quant_fp8"],
            pass_config=PassConfig(fuse_norm_quant=True, eliminate_noops=True),
        ),
    )
    with vllm.config.set_current_vllm_config(vllm_config), monkeypatch.context() as m:
        from vllm.compilation.passes.fusion.rocm_aiter_fusion import
        RocmAiterRMSNormQuantFusionPass
        torch.set_default_device("cuda")
        torch.set_default_dtype(torch.bfloat16)
        torch.manual_seed(0)
        m.setenv("VLLM_ROCM_USE_AITER", "1")
        rocm_aiter_ops.refresh_env_variables()
        fusion_pass = RocmAiterRMSNormQuantFusionPass(vllm_config)
        passes = [NoOpEliminationPass(vllm_config), fusion_pass, PostCleanupPass(vllm_config)]
        backend = TestBackend(passes=passes)
        model = model_cls().eval()
        x = torch.randn(10, HIDDEN_SIZE, dtype=torch.bfloat16, device="cuda")
        result = torch.compile(model, backend=backend, fullgraph=True)(x)
        # 验证 fused 图形中包含正确的融合节点
        # ( 测试后端会检查至少一个 pattern 被替换 )
```

评论区精华

> Rohan138 确认 matcher_utils.py 的更改“good catch, LGTM”。 Rohan138 询问 duplicate quant 的来源，frida-andersson 通过 VLLM_DEBUG_DUMP_PATH 提供证据，确认 DSv3.2 特有。 tjtanaa 要求将图变换门控到 gfx950（“IMPORTANT NOTE: Do not import anything from vllm.platforms.rocm without guarding it with current_platform.is_rocm()”），并要求将 logger.info 降为 logger.debug。 ProExpertProg 建议用声明式 DoubleQuant 模式替代手动图变换（“could we just add a new pattern -> replacement”），避免复杂且脆弱的手写变换。 ChuanLi1101 实施模式并添加 view-tolerant 变体，ProExpertProg 最终批准（“this is in fact much cleaner! Good work”）。 关于 UUID 一致性，tjtanaa 担心移除 clone_elimination.uuid() 影响，ProExpertProg 澄清“removes it from the pass key”，不影响 pass 实际执行。

- gfx950 门控 (gating) (design): ChuanLi1101 在 commit 90474f7 中实施门控，仅在 on_gfx950() 下执行图变换，且 on_gfx950 的导入被包裹在 is_rocm() 检查内。后续模式版本保留相同门控。

- DoubleQuant pattern 替代手动图变换 (design): ChuanLi1101 在 commit 51502209 中移除手动变换，改用 DoubleAiterRMSFp8GroupQuantPattern 声明式模式。ProExpertProg 批准。
- UUID 缓存键一致性 (correctness): 在非 gfx950 平台保持原缓存键，gfx950 上移除 clone_elimination 的 UUID 以避免缓存未命中。
- 日志级别调整 (style): ChuanLi1101 在 commit 317a9eb 中改为 logger.debug。
- view-tolerant 模式验证 (testing): 模式正确匹配，性能提升确认。

风险与影响

- 风险：
  1. matcher_utils.py 全局影响：移除 is_fp8_fnuz() 分支仅影响 match_rocm_aiter=True 路径，但仍可能在非 gfx950 的 ROCm 平台上改变量化算子选择，需确保所有 match_rocm_aiter 用例（gfx942 等）正确调用 get_group_quant_op()。
  2. view-tolerant 模式副作用：引入 view_to_reshape 可能改变图结构，但该函数是 PyTorch 内置转换，已在实际模型上验证。
  3. UUID 缓存键调整：在 gfx950 上移除 clone_elimination.uuid() 可能导致缓存未命中，但此影响已通过仅在 gfx950 上 pop 并确保 non-gfx950 完整键来缓解。
  4. 测试覆盖有限：仅覆盖两种 fan-out 形状，未涵盖其他可能出现 view 的场景（如 AiterFusedAddRMSFp8GroupQuantPattern 的 fan-out），但实际模型已验证无回归。 - 影响：影响范围：仅 ROCm 平台，且仅 AITER 可用且使用 RocmAiterRMSNormQuantFusionPass 的场景（即启用 fuse_norm_quant=True 且 match_rocm_aiter=True 的编译模式）。具体为 gfx950（MI355X）上运行 DeepSeek-V3.2 模型时性能提升约 3.3%（TP4, bf16, HIP graphs）。用户影响：DSv3.2 用户可直接受益；其他模型（Kimi-K2.5 等）融合模式无影响（Replaced 0 patterns）。团队影响：无直接开发负担，但后续应关注其他 ROCm 平台是否出现类似算子选择问题。 - 风险标记：核心编译逻辑变更，gfx950 特定门控，模式匹配依赖图结构，可能影响其他 ROCm 平台（已验证无回归）

关联脉络

- PR #41812 [ROCm][DSv4] implement flash sparse mla with triton kernels: 同属 ROCm 和 DeepSeek 生态的性能优化，涉及 Triton kernel 替换和编译融合，与本 PR 在模块和实验平台上重叠。
- PR #40392 [Performance][DSR1]: Fused RoPE+KVCache+q_concat for MLA: 针对 DeepSeek-R1 的 MLA 融合优化，与本 PR 的 RMSNorm+Quant 融合属于同一条性能提升链路，共享编译融合模式。
- PR #42236 [DSv4] Improved dequant gather K cache kernel: 同样聚焦 DeepSeek 系列模型的性能改进，涉及自定义 kernel 和编译优化，与本 PR 在 DeepSeek 性能线相关。