PR #48757 完整报告

vllm-project/vllm

[Compilation]Fuse Transformers Residual Add + RMSNorm

合并时间：2026-07-30 21:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48757

执行摘要

- 一句话：融合残差加法与 RMSNorm 提升推理吞吐
- 推荐动作：值得精读，特别是 AddRMSNormPattern 和 RMSNormReshapePattern 的实现展示了 vLLM 编译模式匹配和内核融合的典型模式，对于希望扩展编译管道或理解优化 pipeline 的开发者是很好的参考。

功能与动机

Transformers 模型分别发射 residual add 和 RMSNorm，阻碍了 FusedAddRMSNorm 内核的利用。该 PR 将它们规范化为 fused_add_rms_norm，并处理中间的 reshape，使得下游融合（如 AR+RMS、RMS+Quant）得以重新生效。

实现拆解

1. 模式匹配类（add_rms_fusion.py）：定义 AddRMSNormPattern 匹配 add → rms_norm，替换为 fused_add_rms_norm；RMSNormReshapePattern 将 rms_norm → reshape(-1, -1) 转换为先 reshape 再做 norm，以暴露 2D 输入给下游融合；FusedAddRMSNormReshapePattern 类似处理 fused 版本。
2. 内核支持（aiter_ops.py）：修改 _rocm_aiter_rmsnorm2d_fwd_with_add_impl，在调用 AITER 内核前将输入 reshape 为 2D，并将输出 reshape 回原始形状，使 fused 操作支持任意张量形状。
3. Pass 注册（pass_manager.py）：新增 enable_transformers_norm_canonicalization 条件，仅在 Transformers 后端且启用相关融合配置时注册两个新 Pass，并确保执行顺序（AR+RMS 优先于 RMS+Quant）。
4. 单元测试（test_rmsnorm_reshape_fusion.py）：定义 RMSNormModel 和 AddRMSNormModel，验证融合前后输出一致，并检查算子匹配计数。

关键文件：

- vllm/compilation/passes/fusion/add_rms_fusion.py（模块 编译管道；类别 source；类型 core-logic；符号 AddRMSNormPattern, init, pattern, _pattern）：核心新增文件，定义了三个模式匹配类，实现了残差加法与 RMSNorm 的融合逻辑。
- tests/compile/passes/test_rmsnorm_reshape_fusion.py（模块 测试；类别 test；类型 test-coverage；符号 RMSNormModel, init, forward, AddRMSNormModel）：新增测试文件，验证融合 pass 的正确性，包括数值一致性和算子匹配计数。

- vllm/compilation/passes/pass_manager.py（模块 编译管道；类别 source；类型 dependency-wiring）：注册新的融合 Pass，并添加条件判断以仅在 Transformers 后端启用。
- vllm/kernels/aiter_ops.py（模块 内核；类别 source；类型 core-logic）：修改 fused_add_rms_norm 的 ROCm AITER 实现以支持任意形状输入。

关键符号：AddRMSNormPattern.init, AddRMSNormPattern.pattern, AddRMSNormPattern.replacement, AddRMSNormPattern.get_inputs, RMSNormReshapePattern.init, RMSNormReshapePattern.pattern, RMSNormReshapePattern.replacement, RMSNormReshapePattern.get_inputs, AddRMSNormFusionPass.init, RMSNormReshapeFusionPass.init, _rocm_aiter_rmsnorm2d_fwd_with_add_impl, test_rmsnorm_reshape_fusion, test_add_rmsnorm_reshape_fusion, _run_fusion_test

关键源码片段

tests/compile/passes/test_rmsnorm_reshape_fusion.py

新增测试文件，验证融合 pass 的正确性，包括数值一致性和算子匹配计数。

```
# tests/compile/passes/test_rmsnorm_reshape_fusion.py
# Test RMSNormReshapeFusionPass and AddRMSNormFusionPass

import pytest
import torch
import vllm.ir.ops
from tests.compile.backend import TestBackend
from vllm.compilation.passes.fusion.add_rms_fusion import (
    AddRMSNormFusionPass,
    RMSNormReshapeFusionPass,
)
from vllm.compilation.passes.fx_utils import find_op_nodes, is_func
from vllm.compilation.passes.utility.noop_elimination import NoOpEliminationPass
from vllm.compilation.passes.utility.post_cleanup import PostCleanupPass
from vllm.config import CompilationConfig, CompilationMode, VllmConfig
from vllm.platforms import current_platform

pytestmark = pytest.mark.skipif(
    not current_platform.is_cuda_alike(), reason="Requires CUDA or ROCm"
)


class AddRMSNormModel(torch.nn.Module):
    """Simulate Transformers residual add + RMSNorm pattern"""
    def __init__(self, hidden_size: int, residual_first: bool) -> None:
        super().__init__()
        self.weight = torch.nn.Parameter(torch.randn(hidden_size))
        self.residual_first = residual_first
```

```
    def forward(
        self, x: torch.Tensor, residual: torch.Tensor
    ) -> tuple[torch.Tensor, torch.Tensor]:
        # Original graph: add and rms_norm separate
        residual_out = residual + x if self.residual_first else x + residual
        rms = vllm.ir.ops.rms_norm(residual_out, self.weight, 1e-6)
        return rms.reshape(-1, rms.shape[-1]), residual_out

    def ops_in_model_before(self):
        # Before fusion: add and rms_norm
        return [torch.ops.aten.add, torch.ops.vllm_ir.rms_norm]

    def ops_in_model_after(self):
        # After fusion: only fused_add_rms_norm
        return [torch.ops.vllm_ir.fused_add_rms_norm]


@pytest.mark.parametrize("residual_first", [True, False])
def test_add_rmsnorm_reshape_fusion(vllm_config, residual_first):
    add_fusion = AddRMSNormFusionPass(vllm_config)
    reshape_fusion = RMSNormReshapeFusionPass(vllm_config)
    model = AddRMSNormModel(hidden_size=32, residual_first=residual_first)
    x = torch.randn(2, 7, 32)
    residual = torch.randn_like(x)
    backend = _run_fusion_test(
        model, vllm_config, [add_fusion, reshape_fusion], x, residual
    )
    assert add_fusion.matched_count == 1
    assert reshape_fusion.matched_count == 1
    backend.check_before_ops(model.ops_in_model_before())
    backend.check_after_ops(model.ops_in_model_after())
```

评论区精华

- 加法顺序泛化：hmellor 要求支持 residual + branch 和 branch + residual 两种顺序，BadrBasowid 通过 residual_first 参数实现。
- 条件启用：BadrBasowid 建议仅对 Transformers 后端启用，hmellor 指出 model_config.using_transformers_backend() API，最终作为 gating 条件。
- 编译时间：hmellor 测得编译时间增加约 70%，但认为性能收益可接受。
- 融合优先级：讨论确认 AR+RMS 融合优先于 RMS+Quant，在 TP1 场景 RMS+Quant 仍有收益。
- 加法顺序泛化 (design): 添加 residual_first 参数，在构造 AddRMSNormPattern 时指定，测试涵盖两种顺序。
- 条件启用（Gating） (design): 在 pass_manager.py 中添加 enable_transformers_norm_canonicalization 条件，包含 using_transformers_backend() 检查。

- 编译时间增加 (performance): 编译时间增长被接受，作为性能优化的代价。
- 融合优先级 (design): 在 pass 注册顺序中，AR+RMS 先于 RMS+Quant。

风险与影响

- 风险:
  - 编译时间增加：~70% 的编译耗时增长可能影响开发迭代，但在生产部署中以性能优先。
  - 仅 Transformers 后端：通过 gating 限制，非 Transformers 模型不受影响，但需确保 gating 逻辑正确覆盖所有目标模型。
  - 精度风险：内核融合可能引入微小数值差异，已通过 torch.testing.assert_close 验证。
  - 依赖 AITER：ROCm 平台依赖 AITER 内核，若 AITER 不可用则融合退化为原始操作。
  - 回归风险：新模式匹配可能错误替换非目标模式，现有测试覆盖有限模型。
- 影响:
  - 用户收益：Transformers 模型（Qwen、Llama 等）在 ROCm 和 CUDA 上获得 2-18% 吞吐提升，TTFT/TPOT 降低。
  - 部署影响：编译阶段时间增加，但推理性能提升显著。
  - 团队维护：新增 164 行核心代码和 105 行测试，遵循已有编译 Pass 架构，易于理解。
  - 风险标记：编译时间增加，仅 Transformers 后端，依赖 AITER 内核，新增编译 Pass 回归风险

关联脉络

- PR #49937 [ROCm] Add AITER FP8 ViT encoder attention: 同样涉及 AITER 内核集成和编译融合优化，共享 ROCm 平台优化方向。
- PR #50387 [CPU] Bump up CPU kernels to latest version: 体现了整个仓库对内核融合和性能优化的持续投入，本 PR 是其中一环。