PR #42411 完整报告

vllm-project/vllm

[ROCm] Run AITER RMSNorm pad fusion before AR RMS fusion

合并时间：2026-05-13 18:35

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42411

执行摘要

- 一句话：Reorder ROCm pad fusion before AR+RMS fusion
- 推荐动作：值得合入。该 PR 精准修复了 ROCm AITER 融合管道中的调度竞争问题，通过极小的代码改动（+5/-3）获得显著性能提升，且经过充分验证。适合作为编译 pass 优先级设计的参考案例。

功能与动机

对于 GPT-OSS 风格 MoE 模型，fused_add_rms_norm 节点同时被两个 2-op 融合 pass 竞争：RocmAiterTritonAddRMSNormPadFusionPass 匹配更宽的模式（fused_add_rms_norm → router GEMM → pad），而 RocmAiterAllReduceFusionPass 匹配通用模式。Inductor 贪心匹配，先运行的 pass 会抢占节点。原先 AR+RMS 融合先运行，导致 pad 融合在服务编译形状下实际触发 0 次，无法发挥其性能优势。PR 通过交换顺序让 pad 融合优先，使 108 个节点被 pad 融合捕获，仅剩余 3 个由 AR+RMS 融合处理。

实现拆解

1. 调整 pass 注册顺序：在 vllm/compilation/passes/pass_manager.py 的 configure 方法中，将 RocmAiterTritonAddRMSNormPadFusionPass 的注册位置从 fuse_act_padding 条件块（原位于 fuse_allreduce_rms 之后）提前至 fuse_allreduce_rms 之前。
2. 保留原有条件逻辑：RocmAiterTritonAddRMSNormPadFusionPass 仍受 self.pass_config.fuse_act_padding 和 rocm_aiter_ops.is_enabled() 控制，行为语义不变。
3. 删除原位置注册：移除旧位置（位于 fuse_act_quant 条件块之后的重复注册），改为仅在提前位置注册一次，避免重复执行。
4. 无其他代码改动：不涉及内核、编译参数或配置变更，仅影响 Inductor 后 grad pass 的执行顺序。

关键文件：

- vllm/compilation/passes/pass_manager.py（模块 编译器；类别 source；类型 core-logic）：核心改动文件，调整了两个 ROCm AITER 融合 pass 的注册顺序，从 'fuse_allreduce_rms 先、fuse_act_padding 后 ' 改为 'fuse_act_padding 先、fuse_allreduce_rms 后 '，并消除重复注册。

关键符号：PostGradPassManager.configure

关键源码片段

vllm/compilation/passes/pass_manager.py

核心改动文件，调整了两个 ROCm AITER 融合 pass 的注册顺序，从 'fuse_allreduce_rms 先、fuse_act_padding 后 ' 改为 'fuse_act_padding 先、fuse_allreduce_rms 后 '，并消除重复注册。

```
# vllm/compilation/passes/pass_manager.py (conceptually after change)
class PostGradPassManager(VllmInductorPass):
    def configure(self, config: VllmConfig) -> None:
        self.pass_config = config.compilation_config.pass_config
        with set_current_vllm_config(config, check_compile=False):
            # ... 其他 pass 注册 ...

            # 将 pad 融合提前到 AR+RMS 融合之前，
            # 因为两者都消费 fused_add_rms_norm 节点，
            # Inductor 贪心匹配，先注册的 pass 会优先匹配。
            if self.pass_config.fuse_act_padding and rocm_aiter_ops.is_enabled():
                self.passes += [RocmAiterTritonAddRMSNormPadFusionPass(config)]

            if self.pass_config.fuse_allreduce_rms:
                if rocm_aiter_ops.is_enabled():
                    self.passes += [RocmAiterAllReduceFusionPass(config)]
                else:
                    self.passes += [AllReduceFusionPass(config)]

            # ... 后续 pass ... （删除旧位置的重复注册）
```

评论区精华

Review 中 ProExpertProg 提出质疑："are you saying that it's better to do all_reduce → rms_add_pad than allreduce_rms_add → pad? That seems surprising"，作者 akii96 在 Issue 评论中解释了两者的冲突本质：两个 2-op 融合竞争同一个 fused_add_rms_norm 节点，Inductor 贪心匹配，先运行的 pass 获胜。对于 GPT-OSS 模型，pad 融合获取更大性能收益，尤其低 TP 场景。Rohan138、ProExpertProg 和 tjtanaa 最终均批准。

- Pass 竞争和优先级 (design): 确认顺序调整合理，ProExpertProg 随后批准。

风险与影响

- 风险：回归风险低：变更仅调整两个 pass 的执行顺序，不影响内核逻辑或其他编译步骤。实测精度（gsm8k 5-shot）无退化。潜在风险：若未来新增类似竞争关系的 pass，可能需重新评估优先级。性能回退：在非 MoE 模型或不同架构上，顺序调整可能带来不同收益，但 AR+RMS 融合仍能在剩余节点上触发，不会完全失效。
- 影响：用户影响：对使用 ROCm + AITER + GPT-OSS 风格 MoE（如 openai/gpt-oss-120b）的用户，推理吞吐提升约 7.4%，TPOT 降低 5.5%。性能提升主要集中在 pad 融合能覆盖的层，对于没有 router-pad 子图的层，AR+RMS 融合仍正常执行。

兼容性：无破坏性变更，所有现有配置和模型继续工作。

- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #42326 [AMD] skip machete tests for rocm: 同为 ROCm 相关 PR，涉及测试跳过。
- PR #41892 [Bugfix][Quark] Fix W8A8 INT8 garbage outputs on Step-3.5-Flash (and other 3-key fused-MoE Quark exports): 同为 ROCm 平台下 MoE 模型的量化修复，涉及 ROCm 和 MoE 标签。