

PR #55213 完整报告

vllm-project/vllm

[ROCM] [BugFix] Fix AITER MXFP4 ASM-GEMM crash on unfused shared experts

合并时间: 2026-09-10 13:23

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55213>

执行摘要

- 一句话: 修复 AITER MXFP4 ASM-GEMM 因权重缩放形状不满足对齐要求而导致的崩溃, 并修正共享专家融合逻辑。
- 推荐动作: 此 PR 值得精读, 因为它解决了一个阻止特定硬件 / 模型组合运行的崩溃 bug, 并附带了清晰的测试和验证数据。关键设计决策是采用“逐层守卫”而非全局禁用 ASM GEMM, 以及调整融合决策的优先级。这两处改动都体现了对性能与兼容性权衡的细致考虑。建议关注 `process_weights_after_loading` 中的警告日志设计, 以及 `Qwen3NextSparseMoeBlock` 初始化逻辑的重组。

功能与动机

根据 PR body, 动机是修复使用 `amd/Qwen3.8-Flash-Next-Quark-MXFP4` 模型在 MI350x 上的崩溃。问题根源在于共享专家的 `down_proj` 层的 `in_features=640` 导致其 `weight_scale` 形状为 `(2560, 20)`, 其中列维度 20 不能被 8 整除, 违反了 AITER ASM FP4 GEMM 的 `scale-swizzle` 形状假设, 从而触发 `RuntimeError: shape '[80,2,16,1,2,4,1]' is invalid for input of size 25600`。此外, 评论者 `bichowdh` 报告了另一个模型 `Qwen3.8-27B-Quark-AWQ-MXFP4` 在 TP2 下也因行维度不满足 32 的整除性而崩溃, 验证了此 PR 的修复普适性。

实现拆解

- 添加逐层 ASM 形状守卫: 在 `vllm/model_executor/kernels/linear/mx_fp4/aiter.py` 中新增常量 `_ASM_FP4_SCALE_ROW_MULTIPLE (32)` 和 `_ASM_FP4_SCALE_COL_MULTIPLE (8)` 以及函数 `_asm_fp4_scale_swizzle_supported()`。该函数检查 `weight_scale` 是否为 2D 张量且其行、列维度分别能被 32 和 8 整除。在 `AiterMx_fp4LinearKernel.process_weights_after_loading()` 方法中, 如果 `use_asm_gemm` 为真但形状检查失败, 则记录警告并临时将该层的 `use_asm_gemm` 标志设为 `False`, 从而在后续的 `if self.use_asm_gemm:` 分支中跳过 ASM 专用的 `weight_scale` reshape 和 `weight shuffle`, 转而执行 `else` 分支 (转置 `weight_scale`), 最终在 `apply_weights` 中通过 Triton FP4 GEMM 路径执行计算。
- 调整共享专家融合决策顺序: 在 `vllm/model_executor/models/qwen3_next.py` 的 `Qwen3NextSparseMoeBlock.__init__()` 中, 将 `resolve_layer_fused_shared_expert()` 的调用 (决定是否启用融合共享专家) 提前到 `_should_replicate_misaligned_shared_expert()` (决定是否因 TP 不对齐而复制共享专家) 之前。逻辑调整为: 如果融合共享专家被启用, 则直接设置 `self.replicate_shared_expert = False`, 不再进行 TP 对齐检查; 否则, 才执

行原有的对齐检查逻辑。这确保了当 FSE 开启时，TP 对齐问题不再阻止融合。

3. 补充单元测试：在 tests/kernels/quantization/test_rocm_mxfp4.py 中新增三个测试用例，专门测试 _asm_fp4_scale_swizzle_supported 函数在多种对齐 / 未对齐形状下的返回值，以及 AiterMxfp4LinearKernel.process_weights_after_loading() 在遇到未对齐形状时能正确回退到 Triton 路径并转换 weight_scale 形状。

关键文件：

- vllm/model_executor/kernels/linear/mxfp4/aiter.py（模块 AITER 内核；类别 source；类型 core-logic；符号 _asm_fp4_scale_swizzle_supported, AiterMxfp4LinearKernel.process_weights_after_loading）：核心修复文件，新增了 ASM FP4 GEMM 的逐层形状守卫函数和回退逻辑，是解决崩溃问题的关键。
- vllm/model_executor/models/qwen3_next.py（模块 模型结构；类别 source；类型 control-flow；符号 Qwen3NextSparseMoeBlock.init）：修复共享专家融合逻辑，确保在 FSE 启用时不会因 TP 对齐问题而被错误禁用，是解决融合模式下功能问题的关键。
- tests/kernels/quantization/test_rocm_mxfp4.py（模块 MXFP4 测试；类别 test；类型 test-coverage；符号 test_asm_fp4_scale_swizzle_supported_shape_rules, test_asm_fp4_scale_swizzle_rejects_non_2d, test_aiter_mxfp4_process_weights_falls_back_to_triton_for_misaligned_scale）：为新增的形状守卫逻辑提供直接的单元测试覆盖，验证了函数在不同形状下的行为。

关键符号：_asm_fp4_scale_swizzle_supported, AiterMxfp4LinearKernel.process_weights_after_loading, Qwen3NextSparseMoeBlock.init, test_asm_fp4_scale_swizzle_supported_shape_rules, test_asm_fp4_scale_swizzle_rejects_non_2d, test_aiter_mxfp4_process_weights_falls_back_to_triton_for_misaligned_scale

关键源码片段

vllm/model_executor/kernels/linear/mxfp4/aiter.py

核心修复文件，新增了 ASM FP4 GEMM 的逐层形状守卫函数和回退逻辑，是解决崩溃问题的关键。

```
# 在 AiterMxfp4LinearKernel.process_weights_after_loading 方法中添加的守卫逻辑
# 原逻辑: if self.use_asm_gemm: 执行 ASM 专用处理
# 新逻辑: 先检查形状是否支持, 不支持则回退
ASM_FP4_SCALE_ROW_MULTIPLE = 32
ASM_FP4_SCALE_COL_MULTIPLE = 8
```

```
def _asm_fp4_scale_swizzle_supported(weight_scale: torch.Tensor) -> bool:
    """检查 weight_scale 是否满足 ASM swizzle 的对齐要求。"""
    if weight_scale.ndim != 2:
        return False
    sm, sn = weight_scale.shape
    return (
        sm % ASM_FP4_SCALE_ROW_MULTIPLE == 0 and
        sn % ASM_FP4_SCALE_COL_MULTIPLE == 0
    )
```

)

```
def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
    # 如果启用了 ASM GEMM 但当前层的 weight_scale 形状不满足对齐要求
    if self.use_asm_gemm and not _asm_fp4_scale_swizzle_supported(
        layer.weight_scale.data):
        # 记录警告并临时禁用本层的 ASM 路径, 转用 Triton 路径
        logger.warning_once(
            "AITER ASM FP4 GEMM requires weight_scale dims divisible by "
            "(%d, %d), but this layer has weight_scale shape %s. Falling "
            "back to the AITER Triton FP4 GEMM for this layer.",
            ASM_FP4_SCALE_ROW_MULTIPLE, ASM_FP4_SCALE_COL_MULTIPLE,
            tuple(layer.weight_scale.data.shape))
        self.use_asm_gemm = False

    if self.use_asm_gemm:
        # ... 原有的 ASM 专用 weight_scale reshape 和 weight shuffle ...
    else:
        # 原有的 Triton 路径: 转置 weight_scale
        layer.weight_scale = Parameter(
            layer.weight_scale.data.T.contiguous(), requires_grad=False)
```

vllm/model_executor/models/qwen3_next.py

修复共享专家融合逻辑, 确保在 FSE 启用时不会因 TP 对齐问题而被错误禁用, 是解决融合模式下功能问题的关键。

```
# 在 Qwen3NextSparseMoeBlock.__init__ 中调整的逻辑
# 原逻辑: 先检查 TP 对齐, 再决定是否融合
# self.replicate_shared_expert = _should_replicate_misaligned_shared_expert(...)
# if not self.replicate_shared_expert: self.is_fused_shared_expert_enabled = ...

# 新逻辑: 先决定是否融合, 再根据融合结果决定是否检查 TP 对齐
self.is_fused_shared_expert_enabled = False
if config.shared_expert_intermediate_size > 0:
    self.is_fused_shared_expert_enabled = resolve_layer_fused_shared_expert(
        quant_config, prefix, shared_expert_name="shared_expert")

if self.is_fused_shared_expert_enabled:
    # 如果融合启用, 共享专家不再独立存在, 因此无需检查 / 执行 TP 对齐
    self.replicate_shared_expert = False
else:
    # 仅在未融合时检查 TP 对齐, 决定是否复制共享专家
    shared_expert_group_size = get_quark_ocp_mx_group_size(
        quant_config, f"{prefix}.shared_expert.down_proj")
    self.replicate_shared_expert = _should_replicate_misaligned_shared_expert(
        config.shared_expert_intermediate_size, self.tp_size,
        shared_expert_group_size,
        parallel_config.enable_expert_parallel,
        self.is_sequence_parallel)
```

评论区精华

核心讨论集中在测试的硬件兼容性上。审核者 [tjtanaa](#) 指出新增的三个测试用例缺少平台门控标记，要求为每个测试添加 `@pytest.mark.skipif(not on_gfx950(), reason="gfx950 ROCm only")`，以确保这些仅在 gfx950 (MI350x) 上有效的测试不会在其他平台（如 NVIDIA）上失败。作者 [ColinZ22](#) 随后确认并添加了这些标记。这是一个已解决的技术细节，没有遗留的争议。此外，评论者 [bichowdh](#) 提供了另一个崩溃案例，作为数据点支持了 PR 的修复逻辑，但该评论并未引发设计讨论。

- 测试用例的硬件平台门控 (testing): 作者 [ColinZ22](#) 已接受建议并为所有三个测试添加了所需的 `skipif` 标记。

风险与影响

- 风险:
 1. 回归风险（低）：修复是条件性的。仅在 `weight_scale` 形状不满足对齐要求时才触发回退，对于形状对齐的层，行为完全不变。回退路径是现有的 Triton FP4 GEMM，这是一个已验证的稳定路径。
 2. 性能风险（低）：受影响的层（如共享专家 `down_proj`）将从更快的 ASM GEMM 路径切换到 Triton 路径，可能会带来轻微的性能下降。但 PR body 中的 `gsm8k` 评测结果显示性能影响在统计误差范围内（FSE off: 0.9613 vs FSE on: 0.9682），且修复了崩溃问题，收益大于风险。
 3. 逻辑正确性风险（低）：融合决策顺序的调整是关键。原始逻辑可能导致在 FSE 应该启用时，因 TP 对齐问题而错误地禁用了它。调整后的逻辑确保了融合优先，这更符合设计意图。但需要确保所有调用此 `__init__` 的场景（如 `Qwen4ExpSparseMoeBlock`）都被覆盖和测试，由于继承关系，风险可控。
 4. 测试覆盖风险（中）：新增的测试仅覆盖了 `_asm_fp4_scale_swizzle_supported` 函数和 `process_weights_after_loading` 的回退逻辑。对于融合决策顺序调整后，整个 MoE 块在 FSE+TP 组合下的端到端正确性，PR 主要依赖集成测试 (`lm_eval`)，单元测试覆盖不足。
 - 影响：此 PR 影响范围明确且集中。直接影响在 ROCm 平台（特别是 gfx950/MI350x）上使用 AITER MXFP4 量化并包含共享专家（如 Qwen3 系列 MoE 模型）的用户。修复前，这些用户在启动服务时会因权重加载而崩溃，修复后服务可正常启动。对于使用其他量化方案或平台的用户，此 PR 无影响。对团队的影响是增强了 AITER 后端在多样化模型和硬件配置下的鲁棒性，并明确了内核选择的边界条件。
 - 风险标记：核心路径回退 (ASM -> Triton)，MoE 融合逻辑顺序调整，端到端测试覆盖有限

关联脉络

- PR #55887 [ROCm][Bugfix] Support shared KV prefill in AITER attention: 同属 ROCm AITER 后端修复，修复了 AITER 注意力后端在共享 KV 缓存场景下的问题。两者都涉及 AITER 内核在特定边缘情况下的行为修正，体现了对 AITER 集成鲁棒性的持续改进。
- PR #56010 [CI][XPU] Reject CUDA-IPC weight cache on non-CUDA/ROCm platforms: 同属平台兼容性修复，该 PR 在非 CUDA/ROCm 平台（如 XPU）添加了对 IPC 权重缓存的

检查以防止静默错误。本 PR 在 ROCm 平台添加了对 ASM 内核形状的检查，两者都体现了在平台特定优化路径上增加防御性编程的模式。

- PR #52664 [Performance][ROCm] Integrate aiter indexer scoring and top-k kernels into MiniMax-M3 sparse attention path: 同属 ROCm AITER 性能集成 PR，展示了 AITER 库在 vLLM 中的应用广度。本 PR 修复了 AITER 在特定量化模型上的兼容性问题，是对 AITER 生态的持续维护。