PR #46142 完整报告

vllm-project/vllm

[AMD][OCP MX][CI] Fix tests to not dispatch on `UNFUSED_TRITON` backend on MI300, improve w_mxfp4_a_fp8 emulation support

合并时间：2026-06-24 02:25

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46142

执行摘要

- 一句话：修复 AMD OCP MX 测试分派并改进 W-MXFP4 模拟
- 推荐动作：建议 ROCm 和量化模块的维护者仔细阅读本 PR 的核心逻辑变更，特别是 _fp8_quantize_dequantize 的引入和 input_scale 传递方式的调整。测试覆盖的增强值得肯定。后续应跟踪遗留的 TODO 和注释修复（#46142 相关 Issue）。整体变更设计清晰，可以加速 AMD 平台对 OCP MX 量化模型的支持。

功能与动机

OCP MX 测试在 MI300 上错误地分派到了 UNFUSED_TRITON 后端，该后端不进行输入量化，导致测试无法覆盖所需量化逻辑。此外，W-MXFP4-A-FP8 模拟支持不完善，影响 GPT-OSS 模型在 MI300 上的正确运行。详见 PR body 中引用的讨论链接。

实现拆解

1. 在 vllm/model_executor/layers/fused_moe/utils.py 中提取 _fp8_quantize_dequantize 函数，用于 FP8 量化 + 反量化模拟，并在 moe_kernel_quantize_input 中替换原有内联 QDQ 逻辑；同时允许量化模拟路径（quantization_emulation=True 且 quant_dtype 为平台 FP8）调用该函数而非直接报错。
2. 在 vllm/model_executor/layers/fused_moe/oracle/mxfp4.py 的 convert_gpt_oss_weight_to_mxfp4_moe_kernel_format 函数中为 EMULATION 分支添加 w13_input_scale 和 w2_input_scale 参数处理，计算 per-expert scale 并存储最大值到层参数。
3. 在 vllm/model_executor/layers/fused_moe/experts/triton_moe.py 的 TritonExperts.apply 方法中，增加 input_scale 中间变量：模拟模式下仅使用 a1q_scale（忽略 self.a1_scale fallback），非模拟模式保持原有 fallback 逻辑。
4. 在 vllm/model_executor/layers/fused_moe/experts/ocp_mx_emulation_moe.py 中将 _quant_dtype 从硬编码字符串 mxfp8 改为 current_platform.fp8_dtype()，使平台相关的 FP8 dtype 正确传递。
5. 修改测试文件 tests/quantization/test_quark.py 和 tests/models/quantization/test_gpt_oss.py：在非 gfx950 的 AMD GPU 上强制设置 moe_backend='emulation'，确保模拟后端被选中。

6. 在 vllm/model_executor/layers/quantization/quark/quark_moe.py 的 _setup_kernel 方法中调用 convert_gpt_oss_weight_to_mxfp4_moe_kernel_format 时传入 w13_input_scale 和 w2_input_scale，使新参数生效。

关键文件：

- vllm/model_executor/layers/fused_moe/utils.py（模块 MoE 量化；类别 source；类型 data-contract；符号 _fp8_quantize_dequantize）：核心变更：新增 _fp8_quantize_dequantize 函数，统一模拟路径的量化实现。
- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 MoE 路由；类别 source；类型 data-contract）：在 EMULATION 后端分支添加 input_scale 处理，确保 per-expert scale 正确存储。
- vllm/model_executor/layers/fused_moe/experts/triton_moe.py（模块 Triton 专家；类别 source；类型 data-contract）：修正 input_scale 传递逻辑，模拟模式只使用 a1q_scale。
- vllm/model_executor/layers/fused_moe/experts/ocp_mx_emulation_moe.py（模块 MX 模拟；类别 source；类型 data-contract）：修复 _quant_dtype 使用平台 FP8 dtype，而非硬编码。
- tests/quantization/test_quark.py（模块 量化测试；类别 test；类型 test-coverage；符号 on_gfx950）：强制使用 emulation backend 以确保测试正确性。
- tests/models/quantization/test_gpt_oss.py（模块 模型测试；类别 test；类型 test-coverage）：同样强制 emulation backend 用于 GPT-OSS 测试。
- vllm/model_executor/layers/quantization/quark/quark_moe.py（模块 Quark MoE；类别 source；类型 data-contract）：传递 w13_input_scale 和 w2_input_scale 到权重转换函数。

关键符号：_fp8_quantize_dequantize, moe_kernel_quantize_input, convert_gpt_oss_weight_to_mxfp4_moe_kernel_format, OCP_MXQuantizationEmulationTritonExperts.init, TritonExperts.apply, QuarkOCP_MX_MoEMethod._setup_kernel, on_gfx950

关键源码片段

vllm/model_executor/layers/fused_moe/utils.py

核心变更：新增 _fp8_quantize_dequantize 函数，统一模拟路径的量化实现。

```
def _fp8_quantize_dequantize(
    A: torch.Tensor,
    A_scale: torch.Tensor,
):
    '''对输入进行 FP8 量化再反量化，用于模拟不支持原生 FP8 的场景。'''
    qA, qA_scale = ops.scaled_fp8_quant(A, A_scale, use_per_token_if_dynamic=False)
    A = per_tensor_dequantize(qA, qA_scale).to(A.dtype)
    return A, None

def moe_kernel_quantize_input(
    A, A_scale, quant_dtype, per_act_token_quant, block_shape,
```

```
    is_scale_swizzled, ocp_mx_scheme, quantization_emulation, mx_alignment
):
    # 处理 OCP MX scheme 中需要 QDQ 模拟的情况
    if ocp_mx_scheme is not None:
        if ocp_mx_scheme.endswith('a_fp8'):
            # 无需后续量化，直接返回 QDQ 结果
            return _fp8_quantize_dequantize(A, A_scale)
        # 例如 w_mxfp4 等方案无需 QDQ，继续往下走
    if quant_dtype == current_platform.fp8_dtype():
        if quantization_emulation:
            # 通用 FP8 模拟路径
            return _fp8_quantize_dequantize(A, A_scale)
        else:
            # 原生 FP8 路径
            return _fp8_quantize(A, A_scale, per_act_token_quant, block_shape)
    # 其他类型 (int8, nvfp4 等 )
```

vllm/model_executor/layers/fused_moe/experts/triton_moe.py

修正 input_scale 传递逻辑，模拟模式只使用 a1q_scale。

```
# 在 `TritonExperts.apply` 中，决定传入 kernel 的 input_scale:
# 模拟模式（quantization_emulation）只接受 `a1q_scale`（实际为 None，kernel 内部处理）；
# 非模拟模式保留早期 fallback：优先使用 `a1q_scale`，若为 None 则回退到 `self.a1_scale`。
input_scale = (
    a1q_scale
    if self.quantization_emulation
    else (a1q_scale if a1q_scale is not None else self.a1_scale)
)

# 随后在 `_base_w13_fn` 中将 input_scale 传入 kernel 替代原来的表达式。
def _base_w13_fn():
    invoke_fused_moe_triton_kernel(
        hidden_states,
        w1,
        intermediate_cache1,
        input_scale, # 使用新计算的值
        self.w1_scale,
        None,
        sorted_token_ids,
        expert_ids,
        num_tokens_post_padded,
        False,
        top_k_num,
        config,
        compute_type=compute_type,
        use_fp8_w8a8=self.quant_config.use_fp8_w8a8,
        use_int8_w8a8=self.quant_config.use_int8_w8a8,
        use_int8_w8a16=self.quant_config.use_int8_w8a16,
        use_int4_w4a16=self.quant_config.use_int4_w4a16,
```

```
        per_channel_quant=self.per_act_token_quant,
        block_shape=self.block_shape,
        B_bias=self.w1_bias,
    )
```

评论区精华

主要讨论点

- triton_moe.py 中的 TODO：BowenBao 对新增的 TODO 注释提出疑问，认为模拟模式下应直接用 a1q_scale。作者解释该 TODO 用于未来移除 fallback，当前模拟模式下 a1q_scale 始终为 None，行为正确。
- EMULATION 后端 input_scale 处理职责：BowenBao 询问 input_scale 处理是否已在 quark 量化类中完成；作者指出 convert_gpt_oss_weight_to_mxfp4_moe_kernel_format 是统一的后端预处理方法，在此处处理是合理的。
- utils.py 中注释误导：BowenBao 指出原来注释 "After QDQ, we don't need further quantization" 不准确，但 e2e 行为正确。作者同意作为遗留问题留待后续 PR 修复。
- 测试逻辑是否提炼到 oracle：mgoin 建议将测试中强制 emulation backend 的逻辑放进 oracle 作为单一来源；作者与 BowenBao 讨论后认为当前维持测试级别是稳妥做法，未来可考虑默认 opt-in 并发出警告。
  - triton_moe.py 中 TODO 的理解与修正 (design): 当前代码逻辑合理，保留 TODO 以便后续清理。
  - EMULATION 后端 input_scale 处理的职责归属 (design): 确认此位置的处理是必要的，因为 convert_gpt_oss_weight_to_mxfp4_moe_kernel_format 负责后端特定的预处理。
  - utils.py 中注释误导 (correctness): 已识别问题，未修复，待后续 PR。
  - 是否将测试中的 emulation backend 强制逻辑提炼到 oracle (design): 当前保持测试级别，后续可考虑 oracle 中调整。

风险与影响

- 风险：
  1. 模拟路径修改风险：_fp8_quantize_dequantize 的引入改变了 ocp_mx_scheme 和 quantization_emulation 路径的量化行为，若其他代码路径间接依赖原内联 QDQ 的行为可能导致差异。
  2. 平台特定代码：测试逻辑根据 on_gfx950() 选择后端，仅适用于 AMD 平台，若今后其他平台需要类似处理需额外适配。
  3. 待办遗留：utils.py 中注释错误和 triton_moe.py 中的 TODO 未被解决，可能引起后续维护混淆。
  4. 测试覆盖：测试强制使用 emulation backend 可能隐藏了原生后端（UNFUSED_TRITON）的实际问题，需关注原生后端在相关型号上的表现。 - 影响：影响范围：主要影响 AMD GPU（MI250、MI300、MI325）上运行的 OCP MX 量化模型，特别是 GPT-OSS MoE 模型的 W-MXFP4-A-FP8 模拟测试。测试现在能正确执行输入量化模拟，提高了模型正确性验证的有效性。对于非 AMD 平台（如 NVIDIA）没有影响。

团队中关注 ROCm 量化支持的工程师需仔细 review 模拟路径变更。影响程度：中等。虽然是 bugfix，但仅影响特定平台和量化方案，不涉及通用推理路径。

- 风险标记：模拟路径逻辑修改，平台特定代码（AMD），待办注释遗留，测试强制 backend 选择，涉及核心量化函数

关联脉络

- PR #41436 [ROCm] Make moe emulation backend opt-in: 该 PR 使 emulation backend 成为 opt-in，但测试未相应更新，导致本 PR 需要强制设置。
- PR #42120 [Bugfix] ... (introduced double moe_kernel_quantize_input call): 引入了 ocp_mx_emulation_moe.py 中 a13 的重复 quantize_input 调用，本 PR 修复了该问题。
- PR #44667 [Bugfix] Fix double moe_kernel_quantize_input for nvfp4 emulation: 类似问题在 nvfp4 模拟中修复，本 PR 借鉴了相同思路。
- PR #40857 [Bugfix] ... (introduced self.a1_scale fallback): 引入了 deferred static activation quantization 的 fallback，本 PR 需要修正模拟模式下的行为。
- PR #45896 [Bugfix] ... (fix w_mxfp4_a_fp8 emulation): 相关的前置工作，本 PR 在此基础上进一步改进。