PR #52704 完整报告

vllm-project/vllm

[Bugfix][Quantization] Fix OCP MX MoE emulation silently skipping mxfp6 activation QDQ

合并时间：2026-08-20 00:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52704

执行摘要

- 一句话：修复 OCP MX MoE 模拟静默跳过 mxfp6 激活量化
- 推荐动作：值得精读。该 PR 是典型的“数据契约脱节”修复：上层映射表用了 dispatcher 不认识的键，导致功能静默失效。可借鉴两个设计决策：一是把映射提取为模块级纯函数以支持低成本单元测试；二是对未知 scheme 显式抛 NotImplementedError，让未来新增 scheme 立即暴露而非再次静默。建议同步查看 moe_kernel_quantize_input 的分发实现，理解键的契约来源。

功能与动机

PR body 指出模拟“just quietly stops emulating, and accuracy numbers for these schemes come out optimistic”：4 个 mxfp6-activation scheme 的 quant_dtype 都映射为 "mxfp6"，而 moe_kernel_quantize_input 只分发 "mxfp6_e3m2" 与 "mxfp6_e2m3"，于是全部落入兜底 else、以未量化激活运行，且不报任何错误。另有 w_mxfp6_e2m3_a_fp8 未进入任何映射分支，会在 quant_dtype 属性访问时抛 AttributeError。该问题会让 Quark OCP MX 模型在非原生硬件上的精度评估系统性偏高，需要修复。

实现拆解

修复分四步落地：

1. 根因定位（vllm/model_executor/layers/fused_moe/experts/ocp_mx_emulation_moe.py）：OCP_MXQuantizationEmulationTritonExperts.__init__ 中构建的 scheme→dtype 映射把 w_mxfp4_a_mxfp6_e3m2、w_mxfp4_a_mxfp6_e2m3、w_mxfp6_e3m2_a_mxfp6_e3m2、w_mxfp6_e2m3_a_mxfp6_e2m3 统一映射为 "mxfp6"，但 moe_kernel_quantize_input 的分发键只包含 "mxfp6_e3m2" 与 "mxfp6_e2m3"。这些 scheme 全部落入 dispatcher 兜底 else 分支、直接返回 (A, A_scale)，激活从未被伪量化，且没有任何异常，模拟在静默中失效。
2. 重构映射：新增模块级纯函数 activation_quant_dtype(ocp_mx_scheme)，接受枚举或字符串，按尾数格式拆成 mxfp6_e3m2 / mxfp6_e2m3 两个具体分发键；weight-only scheme 返回 None；在 w_mxfp4_a_fp8、w_mxfp6_e3m2_a_fp8 之外补上此前缺失的 w_mxfp6_e2m3_a_fp8 走 current_platform.fp8_dtype()；未知 scheme 抛 NotImplementedError 显式失败。__init__ 简化为一行 self._quant_dtype = activation_quant_dtype(self.ocp_mx_scheme)。

3. 测试配套（tests/kernels/moe/test_ocp_mx_moe.py）：新增参数化单测 test_emulation_activation_quant_dtype_is_dispatchable，遍历全部 OCP_MX_Scheme 枚举，验证 weight-only 返回 None、激活量化 scheme 经 moe_kernel_quantize_input 后输出确实发生改变；新增端到端测试 test_emulation_a_mxfp6_moe_forward_quantizes_activations，构造 w_mxfp4_a_mxfp6_e3m2 与 weight-only w_mxfp4 两组同权重、同激活、同路由的 MoE forward，断言两者输出不再 bit-identical。
4. 配置与部署：无配置、schema 或部署配套改动；行为面变化集中在 w_mxfp6_e2m3_a_fp8（由 AttributeError 变为有效 fp8 模拟）与未知 scheme（由静默直通变为 NotImplementedError）。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/ocp_mx_emulation_moe.py（模块 量化模拟；类别 source；类型 data-contract；符号 activation_quant_dtype, quant_dtype）：修复核心文件：新增 activation_quant_dtype 纯函数承接 scheme→dtype 映射，修复 4 个 mxfp6-activation scheme 静默跳过激活伪量化的问题，并补全 w_mxfp6_e2m3_a_fp8 的 fp8 路径与 NotImplementedError 兜底。
- tests/kernels/moe/test_ocp_mx_moe.py（模块 测试用例；类别 test；类型 test-coverage；符号 test_emulation_activation_quant_dtype_is_dispatchable, test_emulation_a_mxfp6_moe_forward_quantizes_activations）：测试配套：参数化单测覆盖全部 OCP_MX_Scheme 的分发契约，端到端测试证明 mxfp6 激活 QDQ 不再被静默跳过。

关键符号：activation_quant_dtype, test_emulation_activation_quant_dtype_is_dispatchable, test_emulation_a_mxfp6_moe_forward_quantizes_activations

关键源码片段

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

修复核心文件：新增 activation_quant_dtype 纯函数承接 scheme→dtype 映射，修复 4 个 mxfp6-activation scheme 静默跳过激活伪量化的问题，并补全 w_mxfp6_e2m3_a_fp8 的 fp8 路径与 NotImplementedError 兜底。

```
# OCP MX MoE 模拟层的核心映射：把 scheme 换算成
# `moe_kernel_quantize_input` 能够分发的具体 dtype 键。
# 旧实现把 4 个 mxfp6-activation scheme 统一映射成 "mxfp6"，
# 而 dispatcher 只认识 "mxfp6_e3m2" 与 "mxfp6_e2m3"，
# 于是落入兜底 else 分支、激活原样返回，伪量化被静默跳过。
def activation_quant_dtype(
    ocp_mx_scheme: OCP_MX_Scheme | str,
) -> torch.dtype | str | None:
    # weight-only 系列：不对激活做量化，返回 None
    if ocp_mx_scheme in {
        OCP_MX_Scheme.w_mxfp4,
        OCP_MX_Scheme.w_mxfp6_e3m2,
        OCP_MX_Scheme.w_mxfp6_e2m3,
    }:
```

```
        return None
    elif ocp_mx_scheme == OCP_MX_Scheme.w_mxfp4_a_mxfp4:
        return "mxfp4"
    # mxfp6 激活按尾数格式拆成两个分发键，与 dispatcher 对齐
    elif ocp_mx_scheme in {
        OCP_MX_Scheme.w_mxfp4_a_mxfp6_e3m2,
        OCP_MX_Scheme.w_mxfp6_e3m2_a_mxfp6_e3m2,
    }:
        return "mxfp6_e3m2"
    elif ocp_mx_scheme in {
        OCP_MX_Scheme.w_mxfp4_a_mxfp6_e2m3,
        OCP_MX_Scheme.w_mxfp6_e2m3_a_mxfp6_e2m3,
    }:
        return "mxfp6_e2m3"
    # fp8 激活：使用当前平台 fp8 dtype。此前 w_mxfp6_e2m3_a_fp8
    # 未进任何分支，会在 quant_dtype 属性处抛 AttributeError
    elif ocp_mx_scheme in {
        OCP_MX_Scheme.w_mxfp4_a_fp8,
        OCP_MX_Scheme.w_mxfp6_e3m2_a_fp8,
        OCP_MX_Scheme.w_mxfp6_e2m3_a_fp8,
    }:
        return current_platform.fp8_dtype()
    # 显式失败，避免再次出现"模拟静默失效"的未知 scheme
    raise NotImplementedError(
        f"No emulated activation dtype for OCP MX scheme {ocp_mx_scheme}."
        " Please open an issue."
    )
```

tests/kernels/moe/test_ocp_mx_moe.py

测试配套：参数化单测覆盖全部 OCP_MX_Scheme 的分发契约，端到端测试证明 mxfp6 激活 QDQ 不再被静默跳过。

```
# 数据契约测试：每个激活量化 scheme 的 quant_dtype 都必须能被
# `moe_kernel_quantize_input` 分发。dispatcher 的最后一条 else
# 会原样返回输入，因此不认识的名字（如 "mxfp6" 而非 "mxfp6_e3m2"）
# 会静默跳过伪量化——这正是本 PR 修复的缺陷。
@pytest.mark.skipif(not ROCM_AVAILABLE, reason="emulation backend targets ROCm")
@pytest.mark.parametrize("ocp_mx_scheme", list(OCP_MX_Scheme))
def test_emulation_activation_quant_dtype_is_dispatchable(ocp_mx_scheme):
    quant_dtype = activation_quant_dtype(ocp_mx_scheme)

    # weight-only 方案不允许偷偷量化激活
    if "_a_" not in ocp_mx_scheme.value:
        assert quant_dtype is None, "weight-only schemes must not quantize activations"
        return

    a = torch.randn(64, 128, dtype=torch.bfloat16, device="cuda")
    a_scale = torch.ones(1, dtype=torch.float32, device="cuda")
    out, _ = moe_kernel_quantize_input(
```

```
        a, a_scale, quant_dtype, False, None, quantization_emulation=True
    )
    # 只要输出与输入相等，就说明 dispatcher 没有认这个键
    assert not torch.equal(out, a), (
        f"{ocp_mx_scheme.value} -> quant_dtype={quant_dtype!r} left the activation"
        " unquantized; moe_kernel_quantize_input does not dispatch on it"
    )
```

评论区精华

BowenBao 在 review 中确认旧行为："mxfp6" 未被 dispatcher 匹配时不会抛错，直接落入 else 原样返回输入，这正是 bug 静默的原因。fxmarty-amd 指出问题可追溯到 PR #35737，并在 MI350 上验证 Quark wikitext 正确性测试 measured_value 从 10.5317 升至 10.6309，说明原测试 rtol 过宽未能捕获；他还建议后续让 moe_kernel_quantize_input 与所有 Quark / OCP MX 逻辑统一使用 activation quant key。三位 reviewer（fxmarty-amd、BowenBao、AndreasKaratzas）均 approval。

- 旧实现中 "mxfp6" 未被匹配时是否报错 (correctness): 不会报错，而是静默跳过激活伪量化；本 PR 用具体 dtype 键和 NotImplementedError 兜底修复。
- 后续应改用 activation quant key (design): 本 PR 已修复映射，重构建议留作 follow-up。

风险与影响

- 风险：（1）精度结果变化：修复后 mxfp6-activation scheme 的模拟输出与修复前不同（约 9.44% 平均偏移），依赖旧乐观结果的评估、校准和 benchmark 需要重跑；（2）行为变更：w_mxfp6_e2m3_a_fp8 从 AttributeError 变为有效 fp8 模拟路径，未知 scheme 从静默直通变为 NotImplementedError，属于预期内的显式失败；（3）测试仅标记 ROCM_AVAILABLE 运行，CUDA 等平台未纳入 CI，但该模拟路径本身主要面向无原生 OCP MX 支持的设备；（4）改动集中于 OCP_MXQuantizationEmulationTritonExperts 一条分支，原生 OCP MX 与其它量化后端无影响。
- 影响：影响用户：在非原生 OCP MX 硬件上加载 Quark OCP MX MoE 模型的用户会得到更真实的精度数字，模型输出可能变化。影响系统：仅涉及 OCP MX 模拟专家层的 quant_dtype 数据契约，无新增依赖或配置。影响团队：为 AMD/ROCm 量化路线补上一个静默正确性漏洞，并为后续 activation quant key 重构提供了可测试的纯函数基础。
- 风险标记：模拟输出精度变化（约 9.44% 偏移），行为变更：fp8 scheme 由 AttributeError 转为有效执行，新增 NotImplementedError 显式失败路径，测试仅覆盖 ROCm 平台

关联脉络

- PR #35737 OCP MX MoE emulation（fxmarty-amd 指认为问题来源）：fxmarty-amd 在讨论中确认本 PR 修复的映射问题源自 PR #35737 引入的 OCP MX MoE 模拟逻辑。
- PR #52002 [Bugfix] compressed-tensors: restore int8 grouped WNA16 MoE support: 同为量化后端模拟 / 恢复路径的静默正确性修复，映射键与下游逻辑脱节导致错误结果无报错，修复模式相似。