PR #48949 完整报告

vllm-project/vllm

[ROCm][Quark][7/N] Use MXFP4 linear kernel abstraction for `emulation` backend

合并时间：2026-08-01 01:07

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48949

执行摘要

- 一句话：Quark MXFP4 emulation 重构为共享线性内核抽象
- 推荐动作：值得精读，尤其是内核选择框架的扩展方式（创建 mxfp6/ 子包、扩展 register_linear_kernel、按 activation_quant_key 分发）以及 QuarkScheme 与 kernel 抽象的解耦思路。对关注 ROCm 量化、kernel 抽象统一的工程师有参考价值；建议结合 PR#49348 一起看，理解系列重构的演进脉络。

功能与动机

PR body 明确指出：此前 QuarkOCP_MX 硬编码 self.emulate 布尔并内联分发 AITER 自定义 op，重复了内核选择框架已为 FP8/NVFP4/MXFP8 等方案提供的能力。重构目标是标准化 vLLM 代码库中的 mxfp4 linear 后端，消除 quark_ocp_mx.py 专属的代码 / 分发逻辑，使所有 checkpoint 产商（如 compressed-tensors）受益，并服务于后续 PR#46676。

实现拆解

实现拆解

1. 新增 mxfp4 模拟内核：在 vllm/model_executor/kernels/linear/mxfp4/emulation.py 中新增 EmulationMxfp4LinearKernel，继承 MxFp4LinearKernel，将原 quark_ocp_mx.py 的权重反量化 + 激活 QDQ + F.linear 模拟计算迁入，并通过 activation_quant_key 自动派生 QDQ 函数，避免调用方手动指定。
2. 新增 mxfp6 子包：新建 vllm/model_executor/kernels/linear/mxfp6/，包含 base.py（定义 MxFp6LinearLayerConfig 与抽象基类 MxFp6LinearKernel）、emulation.py（EmulationMxfp6LinearKernel）和 __init__.py。设计上把 mxfp6 权重 / 激活格式独立出来，避免与 mxfp4 的 oracle 混用，也为未来原生 mxfp6 硬件内核预留扩展点。
3. 改造内核选择入口：在 vllm/model_executor/kernels/linear/__init__.py 中，init_mxfp4_linear_kernel 增加 activation_quant_key 参数并构造 MxFp4LinearLayerConfig；新增 init_mxfp6_linear_kernel；在 _POSSIBLE_MXFP4_KERNELS 与新增的 _POSSIBLE_MXFP6_KERNELS 中注册 emulation 内核，并让 register_linear_kernel 支持 mxfp6 类型。
4. 重构 QuarkOCP_MX：在 vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py 中删除 self.emulate 布尔与内联模拟分支，改用 _WEIGHT_QUANT_KEY_MAP/_ACTIVATION_QUANT_KEY_MAP 将 spec 映射为

QuantKey；在 create_weights 中按 weight_quant_key 选择 init_mxfp4_linear_kernel 或 init_mxfp6_linear_kernel，apply_weights 统一委托给 ocp_mx_linear。

5. 适配其余 mxfp4 内核与测试：为 aiter.py、marlin.py、humming.py、xpu.py、flashinfer.py 的 can_implement 增加 activation_quant_key 检查（true-W4A4 内核要求 kMxfp4Dynamic，weight-only 内核容忍并告警）；测试配套新增 tests/kernels/quantization/test_mxfp6_kernel_selection.py，扩展 test_mxfp4_kernel_selection.py，覆盖三类内核的接受 / 拒绝矩阵与 init_* 分发行为。

关键文件：

- vllm/model_executor/kernels/linear/mxfp4/emulation.py（模块 模拟内核；类别 source；类型 core-logic；符号 EmulationMxfp4LinearKernel, init, is_supported, can_implement）：新增核心模拟内核，承载原 QuarkOCP_MX 的 emulation 逻辑，是本 PR 的核心实现单元。
- vllm/model_executor/kernels/linear/mxfp6/emulation.py（模块 模拟内核；类别 source；类型 core-logic；符号 EmulationMxfp6LinearKernel, init, is_supported, can_implement）：新增 MXFP6 模拟内核，将 mxfp6 权重 / 激活格式独立处理，避免与 mxfp4 oracle 混淆。
- vllm/model_executor/kernels/linear/mxfp6/base.py（模块 内核抽象；类别 source；类型 data-contract；符号 MxFp6LinearLayerConfig, MxFp6LinearKernel, init, is_supported）：定义 MXFP6 内核抽象基类与配置 dataclass，为未来 mxfp6 硬件内核提供扩展点。
- vllm/model_executor/kernels/linear/__init__.py（模块 内核选择；类别 source；类型 core-logic；符号 init_mxfp4_linear_kernel, init_mxfp6_linear_kernel）：内核选择入口改造，新增 init_mxfp6_linear_kernel 并为 init_mxfp4_linear_kernel 增加 activation_quant_key 参数，注册 emulation 内核。
- vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py（模块 Quark 量化；类别 source；类型 refactor；符号 QuarkOCP_MX, create_weights, process_weights_after_loading, apply_weights）：重构核心目标文件，移除 self.emulate 分支，统一通过内核选择框架分发。
- vllm/model_executor/kernels/linear/mxfp4/aiter.py（模块 AITER 内核；类别 source；类型 data-contract；符号 can_implement, is_supported）：can_implement 收紧为要求 kMxfp4Dynamic，并增加 AITER 缺失时的告警，影响 ROCm 内核选择路径。
- vllm/model_executor/kernels/linear/mxfp4/marlin.py（模块 Marlin 内核；类别 source；类型 data-contract；符号 can_implement）：weight-only 内核的 activation 兼容处理，接受 None/kMxfp4Dynamic 并告警，是内核契约调整的代表。
- tests/kernels/quantization/test_mxfp6_kernel_selection.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_can_implement_is_abstract, test_emulation_kernel_rejects_non_mxfp6_weights, test_emulation_kernel_accepts_any_supported_config, test_emulation_kernel_rejects_non_mxfp4_or_mxfp6_activation）：新增 MXFP6 内核选择测试，覆盖 emulation 内核的接受 / 拒绝矩阵和 init 分发行为。
- tests/kernels/quantization/test_mxfp4_kernel_selection.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_true_w4a4_kernels_accept_dynamic_mxfp4_activation, test_true_w4a4_kernels_reject_unset_activation,

test_true_w4a4_kernels_reject_explicit_non_mxfp4_activation,test_weight_only_kernels_accept_unquantized_or_mxfp4_activation）：扩展 MXFP4 内核选择测试，覆盖 true-W4A4、weight-only、emulation 三类内核的 activation 语义。

- vllm/model_executor/layers/quantization/utils/quant_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 QuantKey）：QuantKey.dtype 类型放宽为 ScalarType，修复 __str__兼容，是内核契约调整的基础。

关键符号：init_mxfp4_linear_kernel, init_mxfp6_linear_kernel, EmulationMxfp4LinearKernel.apply_weights, EmulationMxfp6LinearKernel.apply_weights, QuarkOCP_MX.create_weights, QuarkOCP_MX.apply_weights, AiterMxfp4LinearKernel.can_implement, MarlinMxfp4LinearKernel.can_implement, HummingMxfp4LinearKernel.can_implement, EmulationMxfp4LinearKernel.can_implement, EmulationMxfp6LinearKernel.can_implement

关键源码片段

vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py

重构核心目标文件，移除 self.emulate 分支，统一通过内核选择框架分发。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

from collections.abc import Callable
from fractions import Fraction
from typing import Any

import torch

from vllm.logger import init_logger
from vllm.model_executor.kernels.linear import (
    MxFp4LinearKernel,
    MxFp6LinearKernel,
    init_mxfp4_linear_kernel,
    init_mxfp6_linear_kernel,
)
from vllm.model_executor.layers.quantization.utils.ocp_mx_utils import (
    OCP_MX_BLOCK_SIZE,
)
from vllm.model_executor.layers.quantization.utils.quant_utils import (
    QuantKey,
    kMxfp4Dynamic,
    kMxfp4Static,
    kMxfp6E2M3Dynamic,
    kMxfp6E2M3Static,
    kMxfp6E3M2Dynamic,
    kMxfp6E3M2Static,
)
from vllm.model_executor.parameter import (
```

```
    GroupQuantScaleParameter,
    ModelWeightParameter,
    PackedvLLMParameter,
)
from vllm.model_executor.utils import set_weight_attrs
from vllm.platforms import current_platform

from .quark_scheme import QuarkScheme

logger = init_logger(__name__)

# 将 Quark 的字符串格式映射到统一 QuantKey，供内核选择框架使用
_WEIGHT_QUANT_KEY_MAP: dict[str, QuantKey] = {
    "mxfp4": kMxfp4Static,
    "mxfp6_e3m2": kMxfp6E3M2Static,
    "mxfp6_e2m3": kMxfp6E2M3Static,
}

_ACTIVATION_QUANT_KEY_MAP: dict[str, QuantKey] = {
    "mxfp4": kMxfp4Dynamic,
    "mxfp6_e3m2": kMxfp6E3M2Dynamic,
    "mxfp6_e2m3": kMxfp6E2M3Dynamic,
}



class QuarkOCP_MX(QuarkScheme):
    # 统一保存选出的线性内核实例，类型可以是 mxfp4 或 mxfp6 内核
    ocp_mx_linear: MxFp6LinearKernel | MxFp4LinearKernel

    def create_weights(
        self,
        layer: torch.nn.Module,
        output_partition_sizes: list[int],
        input_size_per_partition: int,
        params_dtype: torch.dtype,
        weight_loader: Callable,
        **kwargs,
    ):
        # ... 创建 weight / weight_scale 参数（省略） ...

        # 根据权重格式选择对应的内核工厂，并传入激活 QuantKey
        if self.weight_quant_key == kMxfp4Static:
            self.ocp_mx_linear = init_mxfp4_linear_kernel(
                activation_quant_key=self.activation_quant_key,
            )
        elif self.weight_quant_key in [kMxfp6E2M3Static, kMxfp6E3M2Static]:
            self.ocp_mx_linear = init_mxfp6_linear_kernel(
                weight_quant_key=self.weight_quant_key,
                activation_quant_key=self.activation_quant_key,
```

```
        )

    def apply_weights(
        self,
        layer: torch.nn.Module,
        x: torch.Tensor,
        bias: torch.Tensor | None = None,
    ) -> torch.Tensor:
        # 统一委托给内核选择框架选出的内核，不再有 emulation 分支
        return self.ocp_mx_linear.apply_weights(layer, x, bias)
```

评论区精华

Review 中围绕三处设计决策展开讨论：

- MXFP6 是否独立成子包：BowenBao 指出 MxFp4LinearLayerConfig 混入 mxfp6 键值不清晰，建议独立 mxfp6/；mgoin 同意“mxfp6 对 w6a6 当然有意义”，最终落地为独立子包。
- kMxfp4Static 激活语义：mgoin 质疑静态 MXFP4 激活的存在意义，fxmarty-amd 解释静态 scale 目前无人使用且精度代价高，已从 can_implement 接受列表中移除。
- QuantKey.dtype 类型注解：mgoin 指出放宽为 ScalarType 会破坏通用消费者（如 torch.empty）的预期，但现有 INT4/INT8 键已违反注解，属于存量债务，接受后待后续 PR 改进。
  - MXFP6 是否应该独立成 mxfp6/ 子包 (design): 创建独立 mxfp6/ 子包，定义 MxFp6LinearKernel 抽象与 EmulationMxfp6LinearKernel，避免与 mxfp4 oracle 混淆。
  - kMxfp4Static activation 的语义 (correctness): 从 can_implement 的接受 key 中移除 kMxfp4Static，weight-only 内核只接受 None 或 kMxfp4Dynamic。
  - QuantKey.dtype 类型注解放宽 (design): 接受当前改动，在代码中留下 TODO，后续再分离存储 dtype 与逻辑 ScalarType。
  - 模型评测覆盖 (testing): PR 内已补充 GSM8K 端到端测试（Qwen3-1.7B-MXFP4）验证数值一致性，eval 配置后续再补。

风险与影响

- 风险：主要风险集中在内核选择与数值一致性：
- 内核选择行为变化：EmulationMxfp4LinearKernel 被追加到 _POSSIBLE_MXFP4_KERNELS 末尾，ROCm 平台在无 AITER 或内核不匹配时会自动回退到模拟内核，而此前 QuarkOCP_MX 直接走 emulate 分支，两者告警与选择路径存在差异，需验证无回归。
- 数值行为一致性：纯重构声明数值不变，但 weight_scale 的 Parameter 包装、dynamic_mxfp4_quant 与 process_weights_after_loading 的交互顺序发生变化，若顺序有误会导致模型精度回归。
- API 变更：init_mxfp4_linear_kernel 增加参数，外部显式调用者虽可兼容，但若依赖旧签名可能需同步；register_linear_kernel 新增 mxfp6 分支，第三方扩展需适配。

- QuantKey.dtype 放宽：__str__ 已做兼容处理，但其他将 dtype 当 torch.dtype 使用的消费方（如 matcher_utils.py）仍可能受影响，目前仅以 TODO 记录。
- 影响：影响范围涉及量化内核选择框架与 ROCm 上的 Quark 量化路径：
- 用户影响：无 API 破坏，但 ROCm 无 AITER 时的内核选择与告警行为会变化；CUDA 平台新增 emulation fallback。
- 系统影响：统一了 MXFP4/MXFP6 内核分发机制，后续新量化方案（如 compressed-tensors MXFP4）可直接复用，减少重复代码。
- 团队影响：消除 quark_ocp_mx.py 中的重复逻辑，降低维护成本；为未来原生 mxfp6 内核接入提供清晰扩展点。
- 风险标记：核心内核选择路径变更，数值行为回归风险，QuantKey.dtype 类型放宽，ROCm 无 AITER 时选择行为变化

关联脉络

- PR #49348 [ROCm][Quark][6/N] Use MXFP4 linear kernel abstraction for aiter backend: 本 PR 明确参照了该 PR 对 aiter 后端的同类重构，是同系列 [ROCm][Quark] 重构的延续。
- PR #36232 [Quark] Introduce OCP MX emulation in quark_ocp_mx.py: 评论中确认 self.emulate 分支由该 PR 引入，本 PR 将其移除并迁移为独立内核。
- PR #46676 Unified MXFP4 linear kernel for compressed-tensors: PR body 提到本重构对 PR#46676 有帮助，是后续受益的统一 checkpoint 产线工作。