PR #41922 完整报告

vllm-project/vllm

[CPU] Add MXFP4 W4A16 MoE support

合并时间：2026-05-18 18:04

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41922

执行摘要

- 一句话：CPU 新增 MXFP4 W4A16 融合 MoE 内核支持
- 推荐动作：值得精读。该 PR 完整演示了如何在 vLLM 模块化 MoE 架构中集成一个新的量化后端（MXFP4），并涉及 C++ 模板泛化、Python 层路由、测试重构等最佳实践。CPUExpertsMxfp4 类的接口实现方式可作为类似后端的参考。

功能与动机

适配 SGLang PR#16775（sgl-project/sglang/pull/16775），将 MXFP4 W4A16 量化能力引入 vLLM CPU 后端，使 openai/gpt-oss-20b 等大规模 MoE 模型能在 CPU 上以 4-bit 精度运行，显著降低内存占用和计算开销。

实现拆解

1. 新增 CPU MXFP4 专家类：在 vllm/model_executor/layers/fused_moe/experts/cpu_moe.py 中添加 CPUExpertsMxfp4，继承 FusedMoEExpertsMonolithic，实现 _supports_activation、_supports_quant_scheme 等接口，并配套 prepare_mxfp4_moe_layer_for_cpu 函数用于 VNNI-prepack 权重和 scales。
2. 后端路由集成：在 vllm/model_executor/layers/fused_moe/oracle/mxfp4.py 中新增 Mxfp4MoeBackend.CPU 枚举、backend_to_kernel_cls 分支、select_mxfp4_moe_backend CPU 识别逻辑以及 mxfp4_round_up_hidden_size_and_intermediate_size 中对齐约束（BLOCK_N=32），并在 make_mxfp4_moe_quant_config 中增加 CPU 支持。
3. C++ 内核泛化：将 csrc/cpu/sgl-kernels/moe_fp8.cpp 中的 fused_experts_fp8_kernel_impl 泛化为 fused_experts_fp_kernel_impl，通过模板参数 packed_t/param_t/is_mxfp4 支持 FP8 和 MXFP4 两种数据类型。新增 MXFP4 专用 kernel 入口，并针对 scale 偏移计算、packed_K 步长等进行条件适配。
4. 辅助数据结构扩展：在 csrc/cpu/sgl-kernels/gemm.h 中新增 CPUAcTMethod 枚举、CPUQuantMethod::MXFP4、convert_scale_packed 函数声明，以及 tinygemm_kernel 对 MXFP4 的重载；在 gemm_fp8.cpp/moe.cpp/moe.h 中增加对应实现。
5. 测试重构与扩展：删除旧测试文件 tests/kernels/moe/test_cpu_fp8_fused_moe.py，新建 tests/kernels/moe/test_cpu_quant_fused_moe.py，统一覆盖 FP8 W8A16 与 MXFP4 W4A16 两种 kernel，包含 MXFP4QuantizeUtil 工具类和多组 shape 参数化测试。

6. 绑定与导出更新：在 csrc/cpu/torch_bindings.cpp 中绑定 convert_scale_packed，在 vllm/_custom_ops.py 中导出 CPUQuantMethod.MXFP4。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/cpu_moe.py（模块 专家模块；类别 source；类型 data-contract；符号 prepare_mxfp4_moe_layer_for_cpu, CPUExpertsMxfp4, init, expects_unquantized_inputs）：核心 Python 模块：新增 CPUExpertsMxfp4 类与 prepare_mxfp4_moe_layer_for_cpu 函数，实现 MoE 专家层接口。
- tests/kernels/moe/test_cpu_quant_fused_moe.py（模块 量化测试；类别 test；类型 test-coverage；符号 _silu_and_mul, _prepack_experts, _block_dequant_weight, ref_w8a16_block_fp8_moe）：新增测试文件，统一覆盖 FP8 W8A16 和 MXFP4 W4A16 两种 MoE 内核，包含精度和性能测试。
- tests/kernels/moe/test_cpu_fp8_fused_moe.py（模块 FP8 测试；类别 test；类型 deletion；符号 _silu_and_mul, _block_dequant_weight, ref_w8a16_block_fp8_moe, _make_fp8_moe_weights）：删除旧的 FP8 专用测试文件，集中维护。
- csrc/cpu/sgl-kernels/moe_fp8.cpp（模块 CPU 内核；类别 source；类型 core-logic；符号 fused_experts_fp_kernel_impl）：C++ 内核泛化：将 FP8 专用 kernel 重构为模板，支持 MXFP4。
- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 MoE 路由；类别 source；类型 data-contract；符号 Mxfp4MoeBackend.CPU, backend_to_kernel_cls, select_mxfp4_moe_backend, mxfp4_round_up_hidden_size_and_intermediate_size）：后端路由：注册 CPU 为合法 MXFP4 后端，实现权重预处理和配置对齐。

关键符号：prepare_mxfp4_moe_layer_for_cpu, CPUExpertsMxfp4.init, CPUExpertsMxfp4.apply, fused_experts_fp_kernel_impl, convert_scale_packed, select_mxfp4_moe_backend, prepare_fp8_moe_layer_for_cpu

关键源码片段

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

核心 Python 模块：新增 CPUExpertsMxfp4 类与 prepare_mxfp4_moe_layer_for_cpu 函数，实现 MoE 专家层接口。

```
# SPDX-License-Identifier: Apache-2.0
"""CPU FP8 W8A16 and MXFP4 W4A16 fused MoE experts."""

import torch
import vllm.model_executor.layers.fused_moe.modular_kernel as mk
from vllm._custom_ops import CPUQuantMethod, fused_experts_cpu
from vllm.model_executor.layers.quantization.utils.quant_utils import kMxfp4Static


def prepare_mxfp4_moe_layer_for_cpu(
    w13: torch.Tensor,
    w2: torch.Tensor,
```

```
    w13_scale: torch.Tensor,
    w2_scale: torch.Tensor,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor, torch.Tensor]:
    """VNNI-prepack MXFP4 MoE weights and repack scales for CPU AMX kernel."""
    # convert_weight_packed 内部会处理 expert 维度，无需逐 expert 循环
    packed_w13 = torch.ops._C.convert_weight_packed(w13)
    packed_w2 = torch.ops._C.convert_weight_packed(w2)
    # scale 也需要 packing 为行主序块格式
    packed_w13_scale = torch.ops._C.convert_scale_packed(w13_scale)
    packed_w2_scale = torch.ops._C.convert_scale_packed(w2_scale)
    return packed_w13, packed_w2, packed_w13_scale, packed_w2_scale


class CPUExpertsMxfp4(mk.FusedMoEExpertsMonolithic):
    """CPU MXFP4 W4A16 monolithic MoE experts."""

    def __init__(
        self,
        moe_config: FusedMoEConfig,
        quant_config: FusedMoEQuantConfig,
    ):
        super().__init__(moe_config, quant_config)

    @property
    def expects_unquantized_inputs(self) -> bool:
        # 内核期望 FP16/BF16 输入，而非已量化数据
        return True

    @staticmethod
    def _supports_current_device() -> bool:
        return current_platform.is_cpu()

    @staticmethod
    def _supports_activation(activation: MoEActivation) -> bool:
        # 支持 SiLU 和 SwiGLU
        return activation in (MoEActivation.SILU, MoEActivation.SWIGLUOAI)

    @staticmethod
    def _supports_quant_scheme(
        weight_key: QuantKey | None,
        activation_key: QuantKey | None,
    ) -> bool:
        SUPPORTED_W_A = [
            (kMxfp4Static, None),
        ]
        return (weight_key, activation_key) in SUPPORTED_W_A

    @staticmethod
    def _supports_routing_method(
```

```
        routing_method: RoutingMethodType,
    ) -> bool:
        return routing_method in (
            RoutingMethodType.EXPERT_CHOICE_ROUTING,
            RoutingMethodType.ROUTED_WITH_WEIGHTS,
        )
```

tests/kernels/moe/test_cpu_quant_fused_moe.py

新增测试文件，统一覆盖 FP8 W8A16 和 MXFP4 W4A16 两种 MoE 内核，包含精度和性能测试。

```
# SPDX-License-Identifier: Apache-2.0
"""Tests for CPU quantized fused MoE kernels (FP8 W8A16 and MXFP4 W4A16)."""

import math
import pytest
import torch
import torch.nn.functional as F

from vllm.platforms import current_platform
from vllm.utils.torch_utils import set_random_seed

if not current_platform.is_cpu():
    pytest.skip("skipping CPU-only tests", allow_module_level=True)

import vllm._custom_ops as ops

if not hasattr(torch.ops._C, "fused_experts_cpu"):
    pytest.skip("fused_experts_cpu op not available", allow_module_level=True)


# MXFP4 量化工具
class MXFP4QuantizeUtil:
    @staticmethod
    def quantize(weight: torch.Tensor) -> tuple[torch.Tensor, torch.Tensor]:
        """量化 FP32 权重为 MXFP4 格式 (W4A16 块量化)。"""
        # 具体实现依赖上游 sgl-kernels
        ...

# FP8 参考实现（沿用原有逻辑）
BLOCK_SIZE = [128, 128]
_FP8_INFO = torch.finfo(torch.float8_e4m3fn)
FP8_SCALE = _FP8_INFO.max
FACTOR_FOR_SCALE = 1e-3


def test_mxfp4_cpu_fused_moe(
    E: int = 8,
    N: int = 256,
```

```
    K: int = 512,
    topk: int = 2,
    num_tokens: int = 64,
):
    """集成测试：生成随机输入，通过 CPU fused_experts_cpu 计算并与 PyTorch 参考对比。"""
    torch.manual_seed(42)
    hidden_states = torch.randn(num_tokens, K, dtype=torch.bfloat16)
    # 构造 MXFP4 量化权重和 scales
    w1, w2, w1_scale, w2_scale = MXFP4QuantizeUtil.generate_mxfp4_weights(E, N, K)
    # 调用 CPU 内核
    output = ops.fused_experts_cpu(
        hidden_states, w1, w2, topk_weights, topk_ids,
        False, ops.CPUQuantMethod.MXFP4,
        w1_scale, w2_scale, None, None, [32, 32],
        None, None, None, None, True)
    # 与参考结果比较（参考实现略）
    assert torch.allclose(output, ref_output, atol=1e-1, rtol=1e-1)
```

评论区精华

- bigPYJ1151 指出 prepare_mxfp4_moe_layer_for_cpu 可直接调用 convert_weight_packed 而无须逐 Expert 循环（已在 commit 中修复，同时优化了原有的 prepare_fp8_moe_layer_for_cpu 函数）。
- gemini-code-assist 建议为 moe.cpp 中复杂的 buffer_size_nbytes 计算添加注释或辅助函数，以提升可维护性；该建议未在最终 commit 中直接反映。
- bigPYJ1151 审核通过，并提醒增加 CI 测试超时到 30 分钟（因测试耗时约 18 分钟接近限值）；超时调整未包含在本 PR 中。
- convert_weight_packed 不需要逐 expert 循环 (performance): 已修复：两个函数均改为整张量调用，删除循环。
- 内核缓冲区大小计算的维护性 (design): 未在最终 commit 中反映，但 reviewer 未强制要求。
- 增加 CI 测试超时到 30 分钟 (testing): 已接受建议，但未在本 PR 中修改配置。

风险与影响

- 风险：
  - FP8 回归风险：内核泛化将 FP8 专用 kernel 重构为模板，可能引入精度或性能退化。新测试文件同时覆盖 FP8 和 MXFP4，但需要确认测试参数与旧测试一致。
  - CPU AMX 指令集依赖：MXFP4 内核依赖 AVX512 与 AMX，在老款 CPU（仅 AVX2）上会 crash 或 fallback 失效，需在运行时检测 _supports_current_device。
  - 内存格式变更：新增 convert_scale_packed 函数改变 scales 内存布局，可能与其他后端（如 GPU）的 scale 处理冲突，但该函数仅 CPU 使用。
  - 测试时间过长：合并后的测试文件包含大量参数化用例，若 CI 超时不扩容可能被截断，影响回归覆盖。
- 影响：

- 用户：CPU 上运行 MoE 模型（如 GPT-OSS-20B）可获得 4-bit 权重量化，内存减半、吞吐提升；仅影响 CPU 用户。
- 系统：新增 CPUQuantMethod.MXFP4 枚举及 sgl-kernels 同步版本，需关注上游变动；MoE 架构的非 CPU 后端无影响。
- 团队：需要维护新的 C++ 内核与测试；建议后续将 FP8 与 MXFP4 内核进一步抽象以降低重复代码。
- 风险标记：CPU AMX 依赖，FP8 回归风险，测试超时风险，SGLang 同步合规

关联脉络

- 暂无明显关联 PR