

PR #51248 完整报告

vllm-project/vllm

[Quantization][Autoround][XPU] Support AutoRound MXFP8 MoE models

合并时间: 2026-08-31 16:37

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51248>

执行摘要

- 一句话: 新增 AutoRound MXFP8 MoE 量化方案, 支持 Qwen3 等模型在 XPU 上运行
- 推荐动作: 值得精读, 尤其是关注量化方案扩展模式的人: 可以学习 "scheme 接口分层 (get_linear_method / get_moe_method) + FusedMoE 基础设施复用 + monkeypatch 隔离 kernel 的单测" 这套组合拳, 它让一个多硬件厂商特性以极低侵入落地。通用用户只需知道 --quantization inc 现在已支持 MXFP8 MoE 即可。

功能与动机

PR body 明确说明目标是 "Adds XPU support for AutoRound-format MXFP8 MoE models"。在此之前, INC (Intel Neural Compressor) 的 AutoRound 支持中, MXFP8 只有线性层方案 (INCMxftp8LinearScheme), MoE 层只有 MXFP4 方案 (INCMxftp4MoEMethod), MXFP8 的 MoE 路径缺失, 导致 Qwen3-30B-A3B-MXFP8-AutoRound 这类 MoE checkpoint 无法以 inc 量化方式加载。PR 同时用 PIQA (acc 0.7905) 和 GSM8K 5-shot (exact_match 0.9075 / 0.9166) 数据论证精度达标; reviewer yiliu30 还要求附上 Qwen3-30B-A3B-Instruct-2507 的全精度参考基线 (预期 0.9 以上) 以佐证量化损失可控。

实现拆解

1. 新增 MXFP8 MoE 量化方法: 在 vllm/model_executor/layers/quantization/inc/schemes/inc_mxfp8_moe.py 中新增 INCMxftp8MoEMethod, 继承 FusedMoEMethodBase。create_weights 按 [1, 32] 的 block 粒度注册 w13_weight、w2_weight (torch.float8_e4m3fn) 与配套 scale (torch.uint8), 并在入口校验 hidden_size 与 intermediate_size_per_partition 可被 MXFP8_BLOCK_SIZE (32) 整除, 不满足直接抛 ValueError; 同时对 scale 参数打上 quant_method == "block" 标记。process_weights_after_loading 调用 convert_to_fp8_moe_kernel_format 将 AutoRound 序列化布局转为 fp8 MoE kernel 期望布局, 再用 replace_parameter 原地替换参数, 最后通过 make_fp8_moe_quant_config 与 make_fp8_moe_kernel 构建执行 kernel; apply 只是委托 kernel。提交 3e43794 专门修复了 post-load 阶段的处理, 说明早期版本存在权重布局转换不完全的问题。
2. 扩展 MXFP8 scheme 入口: inc_mxfp8_scheme.py 的 INCMxftp8Scheme 新增 get_moe_method, 返回 INCMxftp8MoEMethod(layer.moe_config), 与已有的 get_linear_method 形成对称接口。此前该 scheme 只处理线性层, MoE 层会落入非量化路径。

3. 测试与精度验证配套: tests/quantization/test_auto_round.py 增加 qwen3_30b_a3b_mxfp8 配置 fixture 与真实模型参数 INCMModel/Qwen3-30B-A3B-12L-MXFP8-test; 两个新增单测分别用 monkeypatch 隔离 select_mxfp8_moe_backend, 验证路由 (linear 层走 INCMxfp8LinearScheme、mlp 层走 INCMxfp8MoEMethod、gate 不量化) 和权重注册 (形状、dtype、block 标记、非整除抛错)。PR body 给出 Qwen3-30B-A3B-MXFP8-AutoRound 在 PIQA (acc 0.7905) 与 GSM8K 5-shot (exact_match 0.9166) 上的手动评估结果。

关键文件:

- vllm/model_executor/layers/quantization/inc/schemes/inc_mxfp8_moe.py (模块 量化后端; 类别 source; 类型 core-logic; 符号 INCMxfp8MoEMethod, init, create_weights, process_weights_after_loading): 核心新增文件, 定义 INCMxfp8MoEMethod, 包括 block 权重注册、kernel 格式转换与 apply 委托, 是整套 MXFP8 MoE 支持的主体。
- vllm/model_executor/layers/quantization/inc/schemes/inc_mxfp8_scheme.py (模块 量化方案; 类别 source; 类型 core-logic; 符号 get_moe_method): 为 INCMxfp8Scheme 增加 get_moe_method 入口, 使 MXFP8 scheme 能路由 MoE 层到新的 INCMxfp8MoEMethod, 是路由闭环的关键一环。
- tests/quantization/test_auto_round.py (模块 量化测试; 类别 test; 类型 test-coverage; 符号 test_qwen3_30b_a3b_mxfp8_around_routes_to_mxfp8_moe, test_inc_mxfp8_moe_method_registers_block_scaled_weights, fake_select_mxfp8_moe_backend, DummyMoeConfig): 新增 MXFP8 MoE 的配置 fixture、真实模型参数化用例与两个针对性单测, 覆盖路由与权重注册, 是保证该量化方案可维护性的关键配套。

关键符号: INCMxfp8MoEMethod.init, INCMxfp8MoEMethod.create_weights, INCMxfp8MoEMethod.process_weights_after_loading, INCMxfp8MoEMethod.get_fused_moe_quant_config, INCMxfp8MoEMethod.apply, INCMxfp8Scheme.get_moe_method, test_qwen3_30b_a3b_mxfp8_around_routes_to_mxfp8_moe, test_inc_mxfp8_moe_method_registers_block_scaled_weights

关键源码片段

tests/quantization/test_auto_round.py

新增 MXFP8 MoE 的配置 fixture、真实模型参数化用例与两个针对性单测, 覆盖路由与权重注册, 是保证该量化方案可维护性的关键配套。

```
def test_qwen3_30b_a3b_mxfp8_around_routes_to_mxfp8_moe(
    monkeypatch,
) -> None:
    # 用假的后端选择函数替换真实实现,
    # 从而在不加载真实模型和 kernel 的情况下验证路由逻辑
    expected_backend = object()
    expected_experts_cls = object()
    captured = {}

    def fake_select_mxfp8_moe_backend(*, config):
```

```

captured["moe_config"] = config
return expected_backend, expected_experts_cls

monkeypatch.setattr(
    "vllm.model_executor.layers.quantization.inc.schemes."
    "inc_mxfp8_moe.select_mxfp8_moe_backend",
    fake_select_mxfp8_moe_backend,
)

config = make_qwen3_around_config("qwen3_30b_a3b_mxfp8")
layer = object.__new__(RoutedExperts)
layer.moe_config = cast(Any, "moe-config")

# 同一个 config 对不同层类型应产出不同量化方案：
# q_proj 线性层走 MXFP8 线性方案，mlp 走 MXFP8 MoE 方案，
# gate 层按 extra_config 保持 16-bit 不量化
linear_method = config.get_quant_method(
    object.__new__(LinearBase), "model.layers.0.self_attn.q_proj"
)
moe_method = config.get_quant_method(layer, "model.layers.0.mlp")
gate_method = config.get_quant_method(
    object.__new__(LinearBase), "model.layers.0.mlp.gate"
)

from vllm.model_executor.layers.quantization.inc.schemes.inc_mxfp8_moe import (
    INCMxfp8MoEMethod,
)

assert isinstance(linear_method, INCLinearMethod)
assert isinstance(linear_method.scheme, INCMxfp8LinearScheme)
assert isinstance(moe_method, INCMxfp8MoEMethod)
# 后端选择结果应透传，且 moe_config 应与 layer 上的一致
assert moe_method.mxfp8_backend is expected_backend
assert moe_method.experts_cls is expected_experts_cls
assert captured["moe_config"] == layer.moe_config
assert isinstance(gate_method, UnquantizedLinearMethod)

```

评论区精华

review 中核心交锋集中在三处：

- 为什么 `a1_scale / a2_scale` 传 `None`：jikunshang 在 `inc_mxfp8_moe.py` 第 165 行附近发问，jl9876 解释 `AutoRound` 序列化 `MXFP8` 格式采用动态激活量化，checkpoint 只含量化权重与权重缩放、没有静态激活缩放张量，因此传 `None`，激活缩放由 `MXFP8` 后端运行时生成；jikunshang 认可 ("oh make sense.")。
- 构造逻辑冗余问题：jikunshang 质疑某处初始化代码 "necessary?"，并推测该逻辑是 `vLLM` 在线量化所需；jl9876 实测确认冗余后删除 ("I tested it and confirmed that it was redundant, so I've removed it.")。

- 精度基线补充: yiliu30 总体 LGTM, 但要求附上 Qwen3-30B-A3B-Instruct-2507 的 GSM8K 参考结果 (预期 > 0.9) ; 随后 APPROVED。
- a1_scale / a2_scale 为何传 None (design): 双方达成一致, jikunshang 回复 "oh make sense.", 确认动态激活量化设计合理。
- 构造阶段疑似冗余的初始化逻辑 (style): 冗余代码被删除, 最终合入版本更精简。
- 请求补充全精度参考基线 (testing): PR body 已提供量化模型 GSM8K 5-shot exact_match 0.9075 / 0.9166 的结果, yiliu30 最终 APPROVED。

风险与影响

- 风险:
 1. XPU 后端选择未在单测覆盖: INCMx_fp8_MoE_Method.__init__ 调用 select_mxfp8_moe_backend(config=self.moe) 选择实际 kernel 后端与 experts 容器类, 单测用 monkeypatch 完全绕过了该逻辑; 若 XPU 上后端注册不完整, 模型加载会在 __init__ 阶段失败, 且只有真实模型参数化用例能兜底。
 2. 行为变更点: INCMx_fp8_Scheme.get_moe_method 新增后, 所有走 inc 量化且 is_mxfp8 为真的 MoE 层都会从原本的非量化路径切换到新的 MXFP8 路径, 属于隐式行为切换, 依赖 can_handle 的判定准确性。
 3. 格式耦合: a1_scale / a2_scale 硬编码为 None, 仅适配 " 序列化且动态激活 " 的 AutoRound 变体; 若未来接入含静态激活缩放的 checkpoint, 该方法不会读取, 需要另一个 scheme 分支。
 4. 对齐约束限制: hidden_size 与 intermediate_size_per_partition 必须被 32 整除, 个别 MoE 模型 (hidden 尺寸非 32 倍数) 会直接加载失败; 这是有意的防护, 但也是模型兼容性边界。
 5. 回归面小但精度依赖手工评估: +341/-0 不改既有路径, 理论回归面小; 端到端精度仅依赖 PR 内人工评估, 缺少 CI 自动数值断言。- 影响: 用户侧: Intel XPU 与 CUDA 用户现在可以用 --quantization inc 直接加载 AutoRound 产出的 MXFP8 MoE checkpoint (如 Qwen3-30B-A3B), 打通了此前 MXFP8 MoE 无法加载的断层。系统侧: 新增一个 FusedMoEMethodBase 子类与 scheme 接口方法, 完全复用共享的 MXFP8 基础设施 (convert_to_fp8_moe_kernel_format、make_fp8_moe_kernel、select_mxfp8_moe_backend), 不触碰既有 CUDA 路径。团队侧: 为 Intel INC 团队后续扩展其他 AutoRound 格式 (静态激活、不同 block shape) 提供了可复用的模板, 也展示了厂商侧如何以最小侵入接入 vLLM 量化框架。- 风险标记: 行为变更点: MXFP8 MoE 层由非量化切换到量化路径, XPU 后端选择逻辑未在单测中覆盖, 仅支持动态激活的序列化 AutoRound 格式, 尺寸对齐约束限制模型兼容性, 端到端精度依赖手工评估

关联脉络

- PR #53896 [Model] Support Qwen3.8-Flash-Next: 同属 " 量化 + MoE 模型支持 " 演进线, 涉及 fp8.py 量化扩展与新型缓存 / 稀疏架构, 可与本 PR 的 scheme 扩展方式互相参照。

- PR #54160 [Hy4] support Hy4-preview model: 同为大型 MoE 模型 + 专用量化 / 稀疏 kernel 的支持，且均包含厂商侧提交与测试配套，反映 vLLM 近期对多硬件量化模型支持的持续投入。