PR #44025 完整报告

vllm-project/vllm

[compressed-tensors] Asymmetric support for MoE WNA16 marlin

合并时间：2026-06-02 23:51

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44025

执行摘要

- 一句话：为 compressed-tensors MoE WNA16 Marlin 添加非对称量化支持
- 推荐动作：建议阅读此 PR 以了解如何在 Marlin MoE 量化体系中扩展非对称 zero-point 支持。特别是 moe_packed_to_marlin_zero_points 与 moe_awq_to_marlin_zero_points 的对比，体现了不同量化工具包打包格式的差异。

功能与动机

llm-compressor 用户使用 scheme='W4A16_ASYM' 量化 MoE 模型（如 Qwen3.5-122B-A10B）后，在 vLLM 推理时遇到断言错误 'AssertionError: Only symmetric quantization is supported for MoE'（关联 Issue llm-compressor#2628）。此 PR 旨在消除该限制，使 compressed-tensors 路径支持非对称 WNA16 量化 MoE，解锁精度提升。

实现拆解

1. 添加非对称 QuantKey 常量：在 quant_utils.py 中定义 kInt4StaticAsym 和 kInt4Static32Asym，使用 scalar_types.uint4 类型并将 symmetric=False，与非对称 zero-point 信息配合。
2. 注册支持的量化方案：在 marlin_moe.py 的 SUPPORTED_QUANT_SCHEMES 列表中新增上述两个 Asym 键，使得 MarlinExperts 能识别非对称配置。
3. 修改 MoE 方法入口：在 compressed_tensors_moe_wna16_marlin.py 的 __init__ 中移除关于 symmetric 的断言，改用成员属性 self.symmetric 并据此从 WNA16_SUPPORTED_TYPES_MAP 或 WNA16_ZP_SUPPORTED_TYPES_MAP 选择量化类型；将 symmetric 参数传递给 QuantKey 构造；在 get_weight_shape 中增加对零点几张量名称 w13_zp 和 w2_zp 的支持并定义其形状。
4. 新增 zero-point 格式转换函数：在 marlin_utils.py 中新增 moe_packed_to_marlin_zero_points，它针对 compressed-tensors 的标准比特打包（无 AWQ 的交错）模式，使用 unpack_cols 解包后再调用 marlin_zero_points 进行 Marlin 排列，逐 expert 完成转换。
5. 集成到权重处理流程：在 int_wna16.py 的 _process_weights_marlin 函数中，在排列偏置（biases）之前插入 zero-point 排列步骤：当 w13_qzeros 和 w2_qzeros 不为 None 时，调用上述函数转换。所有修改均为新增或调整，未移除原有逻辑。

关键文件：

- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16_marlin.py（模块 量化层；类别 source；类型 data-contract）：核心入口，移除了对称量化断言，引入非对称类型选择，调整了 weight shape 以包含 zero-point 张量。
- vllm/model_executor/layers/quantization/utils/marlin_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 moe_packed_to_marlin_zero_points）：新增 moe_packed_to_marlin_zero_points 函数，是 zero-point 格式转换的核心逻辑。
- vllm/model_executor/layers/fused_moe/oracle/int_wna16.py（模块 MoE 运算；类别 source；类型 data-contract）：在 _process_weights_marlin 中插入 zero-point 排列步骤，将转换函数集成到实际权重处理管道。
- vllm/model_executor/layers/quantization/utils/quant_utils.py（模块 量化工具；类别 source；类型 data-contract）：添加了非对称 QuantKey 常量 kInt4StaticAsym 和 kInt4Static32Asym，供支持性检查使用。
- vllm/model_executor/layers/fused_moe/experts/marlin_moe.py（模块 MoE 专家层；类别 source；类型 data-contract）：在 SUPPORTED_QUANT_SCHEMES 中添加非对称 QuantKey，使 MarlinExperts 能识别非对称方案。

关键符号：moe_packed_to_marlin_zero_points, _process_weights_marlin, get_weight_shape, CompressedTensorsWNA16MarlinMoEMethod.init, MarlinExpertsBase._supports_quant_scheme

关键源码片段

vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16_marlin.py

核心入口，移除了对称量化断言，引入非对称类型选择，调整了 weight shape 以包含 zero-point 张量。

```
# 片段展示关键修改：__init__ 中移除断言，并动态选择量化类型
class CompressedTensorsWNA16MarlinMoEMethod(CompressedTensorsMoEMethod):
    def __init__(
        self,
        weight_quant: QuantizationArgs,
        input_quant: QuantizationArgs | None,
        moe: FusedMoEConfig,
        layer_name: str | None = None,
    ):
        super().__init__(moe)
        self.weight_quant = weight_quant
        self.input_quant = input_quant
        # [!] 不再强制对称，而是保存 symmetric 标志
        self.symmetric = weight_quant.symmetric

        self.num_bits = weight_quant.num_bits
```

```
        self.packed_factor = 32 // weight_quant.num_bits
        self.strategy = weight_quant.strategy
        self.group_size = weight_quant.group_size
        self.actorder = weight_quant.actorder

        # [!] 根据 symmetric 选择量类型映射
        # WNA16_SUPPORTED_TYPES_MAP 用于对称，WNA16_ZP_SUPPORTED_TYPES_MAP
        用于非对称
        self.quant_type = (
            WNA16_SUPPORTED_TYPES_MAP[self.num_bits]
            if self.symmetric
            else WNA16_ZP_SUPPORTED_TYPES_MAP[self.num_bits]
        )
        # ... 其余逻辑相同
        # [!] 创建 QuantKey 时传入 symmetric 参数
        weight_key = QuantKey(self.quant_type, scale, symmetric=self.symmetric)
        self.wna16_backend, self.experts_cls = select_wna16_moe_backend(
            config=self.moe, weight_key=weight_key)
```

vllm/model_executor/layers/quantization/utils/marlin_utils.py

新增 moe_packed_to_marlin_zero_points 函数，是 zero-point 格式转换的核心逻辑。

```
def moe_packed_to_marlin_zero_points(
    q_zp_packed: torch.Tensor,
    size_k: int,
    size_n: int,
    num_bits: int,
    is_a_8bit: bool = False,
):
    """将 compressed-tensors 打包的 zero points 转换为 Marlin 格式。

    与 AWQ 不同，compressed-tensors 使用标准位打包（无交错），
    所以只需解包后直接应用 Marlin 排列。
    """
    num_experts = q_zp_packed.shape[0]
    # 分配输出张量，形状与输入一致
    output = torch.empty(
        (num_experts, q_zp_packed.shape[1], q_zp_packed.shape[2]),
        device=q_zp_packed.device,
        dtype=q_zp_packed.dtype,
    )
    for e in range(num_experts):
        # 解包：将 packed int4 按列展开
        q_zp = unpack_cols(q_zp_packed[e], num_bits, size_k, size_n)
        # Marlin 排列零点的行列重排
        output[e] = marlin_zero_points(q_zp, size_k, size_n, num_bits, is_a_8bit)
    return output
```

评论区精华

该 PR 仅有一位审核者 (mgoin) 快速批准 (LGTM!)，未形成实质技术讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 缺少测试覆盖：未见针对非对称路径的单元测试或集成测试，主要依赖手动验证。若后续其他后端（如 BATCHED_MARLIN）也需支持零点儿时，可能因未覆盖而出错。
  2. zero-point 转换的容错性：moe_packed_to_marlin_zero_points 中假设 unpack_cols 能正确处理所有 group_size（包括 -1/ 无群组）的情况，但未在 patch 中明确验证该边界。
  3. 性能影响：新增的 zero-point 排列会增加推理初始化时的预处理开销，但属于一次性转换，影响可控。
  4. 兼容性：若用户使用对称量化且未提供 zero-point 张量，原有路径不受影响；新代码仅在有 zero-point 时执行转换，向后兼容。 - 影响：用户影响：使用 compressed-tensors 以非对称 W4A16 方案量化的 MoE 模型可以在 vLLM 中正常推理，获得更好的困惑度（如 Qwen3-30B-A3B 的 word_perplexity 从 9.37 降至 9.25）。系统影响：MoE Marlin 权重处理路径增加了对 zero-point 的感知，但修改集中在 compressed-tensors 专属入口，不影响其他量化方法。团队影响：文档需更新以提及非对称 W4A16 支持；后续对 AWQ 等方法的非对称支持可复用类似模式。
- 风险标记：缺少测试覆盖，新增 zero-point 转换路径，可能忽略 group_size=-1 场景

关联脉络

- 暂无明显关联 PR