PR #48918 完整报告

vllm-project/vllm

[CT] Support Humming for WNA16 MoE

合并时间：2026-08-20 00:31

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48918

执行摘要

- 一句话：为 CT WNA16 MoE 开启 Humming 后端，支持子字节位宽
- 推荐动作：值得精读。重点关注三类设计决策：用 Fraction + ceil 处理 sub-byte 打包维度、Humming 支持判定如何与权重 key 数据契约结合、以及 torch.dtype 与 ScalarType 统一字符串化的处理方式。同时值得学习 review 中从 mock 测试演进到 e2e 测试的过程。

功能与动机

PR body 明确提出目标：Enable Humming for compressed-tensors WNA16 MoE checkpoints。此前这类 checkpoint 由 WNA16 MoE 的 Marlin 路径独占，非 4/8 bit 的子字节位宽（如 AutoRound 产出的 2 bit MLP 权重）会在 Marlin 路径直接 hard-fail。本 PR 将 WNA16 MoE 的 Humming 桥接扩展为接受 compressed-tensors 的 QuantizationArgs，允许 Humming 路径初始化 sub-byte 检查点，并使用 Humming 期望的 MoE 权重布局。作者在 Qwen3-30B-A3B-Instruct-2507-Attn4bits-Mlp2bits-AutoRound 模型上验证 gsm8k 5-shot exact_match 达到 0.909。

实现拆解

1. 打包因子与打包维度改造：在 compressed_tensors_moe_wna16.py 中把 packed_factor 从整数除法改为 Fraction(32, num_bits)，并新增 _packed_dim(dim) 用 ceil(dim * num_bits / 32) 计算打包维度；get_weight_shape 的 Flashinfer / Marlin 两套 shape 全部改走 _packed_dim，保证 4/8 bit 行为不变的同时支持 2/3/5/6/7 bit。
2. 量化参数与 scale 描述放宽：__init__ 中非 4/8 bit 位宽不再抛 ValueError，而是构造统一的 ScaleDesc；group_size == -1 映射为 per-channel，否则为 (1, group_size) 分组。同时 is_transposed 将 Humming 与 Flashinfer 一并排除，因为 Humming 期望非转置权重布局。
3. Humming 支持判定扩展：fused_humming_moe.py 新增 _is_supported_wna16_weight_key，要求权重 key 无二级 scale、scale 分组形状为 per-channel 或 (1, col)、dtype 为 2-8 bit 无符号整数（含 INT4_DTYPE、INT8_DTYPE、uint8）；_supports_quant_scheme 在激活 key 为 None 或 FP8 动态 token 时允许这些 WNA16 key 通过。
4. oracle schema 与字符串化配套：int_wna16.py 的 _humming_wna16_weight_schema 支持 QuantizationArgs，输出 compressed-tensors 格式的 checkpoint schema；quant_utils.py 新增 _dtype_abbr，让 QuantKey / ScaleDesc 的 str() 同时支持

torch.dtype 与 ScalarType。

5. 测试与脚本：tests/quantization/test_moe_wna16.py 新增 CUDA-only 的 create_weights 打包形状测试（3/5/6/7 bit 参数化）与 CT→Humming 转换及 kernel 设置测试；tests/quantization/test_compressed_tensors.py 新增 WNA16 key 支持判定与 QuantKey 字符串格式化测试；另附一个离线推理复现 helper 脚本。

关键文件：

- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py（模块 MoE 量化；类别 source；类型 data-contract；符号 _packed_dim, get_weight_shape, get_fused_moe_quant_config）：核心数据契约改动：打包因子从整数除法改为 Fraction，新增 _packed_dim 以 ceil 计算子字节位宽打包维度，解除仅 4/8 bit 限制，并让 Humming 后端参与权重布局与量化配置转发。
- vllm/model_executor/layers/fused_moe/experts/fused_humming_moe.py（模块 专家层；类别 source；类型 data-contract；符号 _is_supported_wna16_weight_key, _supports_quant_scheme）：Humming 专家层的支持判定扩展：新增 _is_supported_wna16_weight_key，使 _supports_quant_scheme 能识别 2-8 bit 无符号整数的 WNA16 权重 key，是子字节 checkpoint 能进入 Humming 路径的开关。
- vllm/model_executor/layers/quantization/utils/quant_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 _dtype_abbr, QuantKey.str, ScaleDesc.str）：统一 QuantKey/ScaleDesc 字符串化：新增 _dtype_abbr 同时支持 torch.dtype 与 ScalarType，修复 ScalarType 在 str(QuantKey) 中的可读性，这是调试与日志匹配的基础设施。
- tests/quantization/test_moe_wna16.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_compressed_tensors_wna16_moe_create_weights_uses_ceil_packed_shapes, test_compressed_tensors_wna16_moe_converts_and_sets_up_humming_kernel）：新增 W2A16 Humming 回归测试：参数化 3/5/6/7 bit 验证 create_weights 使用 ceil 打包 shape，并覆盖 CT→Humming 转换与 kernel 设置路径，是本次回归覆盖的主战场。
- vllm/model_executor/layers/fused_moe/oracle/int_wna16.py（模块 后端选择；类别 source；类型 data-contract；符号 _humming_wna16_weight_schema）：后端 oracle 的 schema 生成扩展：_humming_wna16_weight_schema 新增 QuantizationArgs 分支，把 compressed-tensors 量化参数映射为 Humming checkpoint schema，否则会抛 TypeError。
- tests/quantization/test_compressed_tensors.py（模块 压缩张量；类别 test；类型 test-coverage；符号 test_humming_supports_compressed_tensors_wna16_quant_key, test_quant_key_str_supports_scalar_type_dtypes）：补充 Humming 对 CT WNA16 QuantKey 的支持判定测试与 QuantKey 字符串格式化测试，验证 _is_supported_wna16_weight_key 和 _dtype_abbr 的行为。
- vllm/model_executor/layers/quantization/utils/humming_utils.py（模块 量化工具；类别 source；类型 data-contract）：Humming 量化配置工具的配套小改，配合 get_fused_moe_quant_config 的转发逻辑。

关键符号：_packed_dim, _is_supported_wna16_weight_key, _dtype_abbr, get_fused_moe_quant_config, _humming_wna16_weight_schema,

test_compressed_tensors_wna16_moe_create_weights_uses_ceil_packed_shapes,test_compressed_tensors_wna16_moe_converts_and_sets_up_humming_kernel

关键源码片段

vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16.py

核心数据契约改动：打包因子从整数除法改为 Fraction，新增 _packed_dim 以 ceil 计算子字节位宽打包维度，解除仅 4/8 bit 限制，并让 Humming 后端参与权重布局与量化配置转发。

```
# 打包因子改用 Fraction：sub-byte 位宽（如 2/3/5/6/7 bit）无法被 32 整除，
# 整数除法会丢失精度，Fraction 可保留精确的 32/N 关系供后续 ceil 计算使用。
self.packed_factor = Fraction(32, weight_quant.num_bits)

# 4/8 bit 沿用既有 scale 常量；其余位宽构造统一 ScaleDesc，
# group_size == -1 表示 per-channel，否则为 (1, group_size) 的分组 scale。
if self.num_bits == 4:
    if self.group_size == 32:
        scale = kInt4Static32GroupScale
    else:
        scale = kInt4StaticGroupScale
elif self.num_bits == 8:
    scale = kInt8StaticGroupScale
else:
    scale = ScaleDesc(
        dtype=torch.float16,
        static=True,
        group_shape=(
            GroupShape.PER_CHANNEL
            if self.group_size == -1
            else GroupShape(row=1, col=self.group_size)
        ),
    )

# Humming 与 Flashinfer 一样使用非转置权重布局，因此 is_transposed 需要排除 Humming，
# 否则后续权重预处理会按转置语义 reshape，导致 shape 不匹配。
self.is_transposed = self.wna16_backend not in (
    WNA16MoEBackend.FLASHINFER_TRTLLM,
    WNA16MoEBackend.HUMMING,
)

def _packed_dim(self, dim: int) -> int:
    # 打包维度 = ceil(dim * num_bits / 32)：不足一个 int32 单元的余数也占位，
    # 与 dense compressed-tensors 的 pack 行为保持一致。
    return math.ceil(dim * self.num_bits / 32)
```

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

Humming 专家层的支持判定扩展：新增 _is_supported_wna16_weight_key，使 _supports_quant_scheme 能识别 2-8 bit 无符号整数的 WNA16 权重 key，是子字节 checkpoint 能进入 Humming 路径的开关。

```
def _is_supported_wna16_weight_key(weight_key: QuantKey | None) -> bool:
    # 仅接受单层 scale；带 scale2 的量化类型（如 MXFP）不适用该 WNA16 判定。
    if weight_key is None or weight_key.scale2 is not None:
        return False

    # scale 只能按通道或按 (1, col) 分组，逐 token 或逐张量 scale 不支持。
    group_shape = weight_key.scale.group_shape
    if not (
        group_shape == GroupShape.PER_CHANNEL
        or (group_shape.row == 1 and group_shape.col > 0)
    ):
        return False

    # 内建 INT4 / INT8 ScalarType 与 torch.uint8 直接放行。
    dtype = weight_key.dtype
    if dtype in (INT4_DTYPE, INT8_DTYPE, torch.uint8):
        return True

    # 其余情况要求 ScalarType：无符号整数，位宽落在 2-8 bit，
    # 覆盖 compressed-tensors 子字节 checkpoint 的实际 dtype。
    return (
        isinstance(dtype, ScalarType)
        and dtype.is_integer()
        and not dtype.is_signed()
        and 2 <= dtype.size_bits <= 8
    )


# _supports_quant_scheme 末尾的判定：激活侧 Humming 会推迟输入量化（见
# expects_unquantized_inputs），因此只要激活 key 为 None 或 FP8 动态 token，
# 且权重 key 满足 WNA16 条件即可放行，不再要求精确命中 SUPPORTED_W_A 表格。
return (weight_key, activation_key) in SUPPORTED_W_A or (
    activation_key in (None, kFp8DynamicTokenSym)
    and _is_supported_wna16_weight_key(weight_key)
)
```

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

统一 QuantKey/ScaleDesc 字符串化：新增 _dtype_abbr 同时支持 torch.dtype 与 ScalarType，修复 ScalarType 在 str(QuantKey) 中的可读性，这是调试与日志匹配的基础设施。

```
def _dtype_abbr(dtype: torch.dtype | ScalarType) -> str:
    # torch.dtype 走 fx.graph.dtype_abbrs 的短名表；ScalarType 直接取 str，
    # 统一 QuantKey / ScaleDesc 的 str() 输出，避免 ScalarType 落入 dtype_abbrs 的 KeyError。
    if isinstance(dtype, ScalarType):
        return str(dtype)
```

```
    return fx.graph.dtype_abbrs[dtype]
```

评论区精华

review 的核心讨论集中在三点：一是 mgoin 指出打包 shape 必须对齐 dense compressed-tensors 的 Fraction/ceil 实现，作者最终落地为 Fraction(32, num_bits) 与 _packed_dim；二是 mgoin 指出 get_fused_moe_quant_config 没有转发 swiglu_alpha / swiglu_beta / swiglu_limit，参考 AWQ/GPTQ/MoeWNA16 已有 Humming 集成，作者回复 Updated；三是 mgoin 认为初始测试只是 private helpers 加 mock 选择，建议做更端到端的 create_weights 与 CT→Humming 转换→kernel 路径测试，作者随后把测试迁移到 tests/quantization/test_moe_wna16.py 并补齐两个 e2e 测试。此外 hshen14 质疑 group_size 应为非负，yiliu30 解释 -1 表示 per-channel 量化属合法值；HDCharles 对 QuantKey.__str__ 的 dtype 分支不解，mgoin 解释只是提取公共 helper，作者确认行为不变。

- 打包 shape 需与 dense compressed-tensors 对齐（Fraction/ceil） (correctness): 作者采用 Fraction(32, num_bits) 与 _packed_dim 的 ceil 实现，并在测试中覆盖 3/5/6/7 bit 验证打包 shape。
- Humming 量化配置需转发 swiglu 参数 (correctness): 作者回复 Updated，最终通过 get_humming_moe_quant_config(layer) 接入 Humming 量化配置。
- 测试应更端到端而非 mock 选择 (testing): 作者将 CT WNA16 MoE 测试迁移至 test_moe_wna16.py，新增 create_weights 打包 shape 与 kernel 设置两个 e2e 测试。
- group_size == -1 是否合法输入 (question): 保持 -1 语义，不添加断言。
- QuantKey.__str__的 dtype 分支提取 (design): 提取公共 helper，行为不变，ScalarType 与 torch.dtype 输出统一。

风险与影响

- 风险：
  - 打包布局回归风险：get_weight_shape 的 Marlin / Flashinfer 分支同样改用 _packed_dim。对 4/8 bit，ceil 与整除结果等价；对非整除位宽，非 Humming 后端若被选中会生成新 shape，需要关注旧 checkpoint 的兼容性。
  - 转置语义调整：is_transposed 对 Humming 变为 False，权重预处理不再按转置语义 reshape，依赖 Humming kernel 的非转置布局契约，若布局推断有误会导致 shape 不匹配。
  - 量化校验放宽：去掉了 num_bits 8 的 group_size 断言，并把任意 2-8 bit 无符号整数纳入支持，错误可能从加载期推迟到 kernel 初始化期，错误信息变晚。
  - 外部依赖：Humming 后端强依赖 nvidia 的 humming 库与 CUDA，测试通过 importorskip 与 skipif 保护，CI 覆盖有限，非 NVIDIA 平台无法验证此路径。
  - 上游演进：与 #49610 Humming refactor 存在同步关系，mgoin 在 review 中指出 prepare_humming_layer 的 AttributeError 失败，主分支继续重构时本路径可能再次被破坏。
  - 影响：对用户：支持 2/3/5/6/7 bit 子字节 WNA16 compressed-tensors MoE checkpoint（如 AutoRound 2 bit 模型）在 Humming 后端运行，扩大了可用量化模型

面；对 4/8 bit 现有模型行为保持不变。对系统：WNA16 后端选择 oracle、打包布局与转置语义三处数据契约变更，影响 Marlin / Flashinfer / Humming 共享的权重创建逻辑。对团队：quantization 与 MoE 后端维护者需要关注 _packed_dim、_is_supported_wna16_weight_key 和 QuantKey 字符串化改动；测试向 test_moe_wna16.py 聚集，形成 WNA16 回归测试主阵地。

- 风险标记：子字节打包布局变更，后端选择与转置语义调整，强依赖外部 Humming 库，量化校验放宽

关联脉络

- PR #49610 Humming refactor: mgoin 在 review 中明确要求本 PR 跟随 Humming refactor 更新 prepare_humming_layer 调用，两个 PR 存在直接依赖与合并冲突关系。
- PR #52002 [Bugfix] compressed-tensors: restore int8 grouped WNA16 MoE support: 同属 compressed-tensors WNA16 MoE 功能线，涉及同一核心文件，此前刚恢复 int8 分组支持，本 PR 继续扩展子字节位宽与 Humming 后端。