PR #48451 完整报告

vllm-project/vllm

[feature]Add int4 quantization support for emulation moe backend

合并时间：2026-07-16 10:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48451

执行摘要

- 一句话：为 ROCm 添加 int4 量化模拟 MoE 后端
- 推荐动作：值得精读，尤其是 Int4EmulationTritonExperts 的反量化设计模式（加载时一次性转换，前向保持纯浮点），以及如何无侵入地在现有路由系统中插入新后端。对需要在无原生 int4 内核平台上运行量化模型的场景有参考价值。

功能与动机

ROCm 平台缺乏支持非对称 int4 量化的 MoE 后端，因此需要模拟后端以运行 int4 量化模型（如 cyankiwi/MiniMax-M3-AWQ-INT4），见 PR body 描述。

实现拆解

1. 新增专家类：vllm/model_executor/layers/fused_moe/experts/int4_emulation_moe.py 中定义 Int4EmulationTritonExperts，继承 TritonExperts，在 __init__ 中清除 quant_config 中的 int4 dtype 和 scale 字段，使基类认为权重是浮点；apply() 中校验权重类型并直接调用父类方法。
2. 扩展 Oracle 层：vllm/model_executor/layers/fused_moe/oracle/int_wna16.py 中新增 WNA16MoEBackend.EMULATION 枚举，并在 backend_to_kernel_cls、_get_priority_backends、map_wna16_backend、make_wna16_moe_kernel 等关键路由函数中串联 emulation 后端；同时实现 _unpack_and_dequant_int4_gptq、_unpack_and_dequant_int4_awq 等反量化辅助函数供 weight loading 调用。
3. 压缩张量方法适配：在 compressed_tensors_moe.py 中拦截 moe_backend == "emulation" 分支，路由到 CompressedTensorsWNA16MarlinMoEMethod；并在 compressed_tensors_moe_wna16_marlin.py 中调整 process_weights_after_loading 以跳过 Marlin 专属的 workspace 分配、增加 g_idx 等参数的 None 保护。
4. 配置扩展：config.py 中为 int4_w4a16_moe_quant_config 和 int8_w8a16_moe_quant_config 增加 gemm1_clamp_limit/alpha/beta 参数，支持 SWIGLU 的 clamp 配置。
5. 测试配套：新增 tests/kernels/quantization/test_int4_emulation_moe.py，包含反量化单元测试（对照参考实现）和端到端 MoE 前向测试，覆盖对称 / 非对称量化以及专家并行（EP）场景。

关键文件：

- vllm/model_executor/layers/fused_moe/experts/int4_emulation_moe.py（模块 MoE 层；类别 source；类型 core-logic；符号 Int4EmulationTritonExperts, init, _supports_current_device, _supports_quant_scheme）：核心新文件，定义 Int4EmulationTritonExperts 类，实现 int4 反量化和前向转发。
- vllm/model_executor/layers/fused_moe/oracle/int_wna16.py（模块 MoE 层；类别 source；类型 core-logic；符号 _unpack_and_dequant_int4_gptq, _unpack_and_dequant_int4_awq, _process_weights_emulation_gptq, _process_weights_emulation_awq）：核心路由文件，新增 EMULATION 后端枚举、反量化辅助函数以及后端选择逻辑。
- tests/kernels/quantization/test_int4_emulation_moe.py（模块 测试；类别 test；类型 test-coverage；符号 _quantize_sym, _quantize_asym, _dequantize_ref, _pack_gptq_zeros）：新增完整测试套件，覆盖反量化单元测试和端到端前向测试（含 EP）。
- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe_wna16_marlin.py（模块 量化；类别 source；类型 data-contract）：修改 process_weights_after_loading 以跳过 Marlin 专属步骤（workspace、g_idx 保护），适配 emulation 后端。
- vllm/model_executor/layers/quantization/compressed_tensors/compressed_tensors_moe/compressed_tensors_moe.py（模块 量化；类别 source；类型 data-contract）：新增 emulation 后端路由分支，串联 CompressedTensorsWNA16MarlinMoEMethod。
- vllm/model_executor/layers/fused_moe/config.py（模块 配置；类别 source；类型 data-contract）：为 int4 和 int8 量化配置添加 gemm1_clamp_limit/alpha/beta 参数，支持 SWIGLU 的 clamp 配置。

关键符号：Int4EmulationTritonExperts.init, Int4EmulationTritonExperts.apply, Int4EmulationTritonExperts._supports_quant_scheme, _unpack_and_dequant_int4_gptq, _unpack_and_dequant_int4_awq, _process_weights_emulation_gptq, _process_weights_emulation_awq, backend_to_kernel_cls, map_wna16_backend, make_wna16_moe_kernel

关键源码片段

vllm/model_executor/layers/fused_moe/oracle/int_wna16.py

核心路由文件，新增 EMULATION 后端枚举、反量化辅助函数以及后端选择逻辑。

```
# 以下展示 WNA16MoEBackend 枚举新增 EMULATION 以及 backend_to_kernel_cls 中的分发逻辑

class WNA16MoEBackend(Enum):
    MARLIN = "MARLIN"
    BATCHED_MARLIN = "BATCHED_MARLIN"
    HUMMING = "HUMMING"
    CPU = "CPU"
    FLASHINFER_TRTLLM = "FLASHINFER_TRTLLM"
    XPU = "XPU"
    EMULATION = "EMULATION" # 新增枚举值
```

```
def backend_to_kernel_cls(
    backend: WNA16MoEBackend,
) -> list[type[mk.FusedMoEExperts]]:
    if backend == WNA16MoEBackend.EMULATION:
        from vllm.model_executor.layers.fused_moe.experts.int4_emulation_moe import (
            Int4EmulationTritonExperts,
        )
        return [Int4EmulationTritonExperts]
    # 其他后端保持不变 ...

# 反量化辅助函数示例：GPTQ 格式 unpack+dequant

def _unpack_and_dequant_int4_gptq(
    w_int32: torch.Tensor, # [n_groups, N//8, 1] int32 — packed 权重
    scale: torch.Tensor, # [n_groups, N] float — 每组缩放因子
    qzeros: torch.Tensor, # [n_groups, N//8] int32 — packed 零点
) -> torch.Tensor:
    """Unpack GPTQ int4 权重并反量化为 BF16."""
    # 实际实现略，流程：解包 -> 应用零点 -> 乘以 scale -> 返回 BF16
    ...
```

tests/kernels/quantization/test_int4_emulation_moe.py

新增完整测试套件，覆盖反量化单元测试和端到端前向测试（含 EP）。

```
# 测试反量化函数 _unpack_and_dequant_int4_gptq 的正确性

import torch
import pytest
from vllm.model_executor.layers.fused_moe.oracle.int_wna16 import (
    _unpack_and_dequant_int4_gptq,
)

def _dequantize_ref(w_uint, scale, zero=None, dtype=torch.bfloat16):
    """参考反量化：手动计算 (unpacked_weights - zero) * scale"""
    K, N = w_uint.shape
    n_groups = scale.shape[0]
    group_size = K // n_groups
    w = w_uint.reshape(n_groups, group_size, N).to(dtype)
    s = scale.unsqueeze(1).to(dtype)
    if zero is None:
        return ((w - 8) * s).reshape(K, N)
    z = zero.unsqueeze(1).to(dtype)
    return ((w - z) * s).reshape(K, N)


class TestUnpackDequantGPTQ:
    @pytest.mark.parametrize("E,K,N,group_size", [
        (2, 64, 32, 32),
        (4, 128, 64, 64),
```

```
])
def test_gptq(self, E, K, N, group_size):
    # 构造随机 int4 权重并打包
    w_fp = torch.randn(E, K, N, dtype=torch.bfloat16, device="cuda")
    # 模拟 GPTQ 打包 + 反量化 ...
    result = _unpack_and_dequant_int4_gptq(packed, scale, zeros)
    expected = _dequantize_ref(unpacked, scale, zeros)
    torch.testing.assert_close(result, expected, atol=1e-2, rtol=1e-2)
```

评论区精华

- 测试代码风格：tjtanaa 要求遵循函数式测试而非类风格，qli88 重构为纯函数。
- 变量命名：tjtanaa 建议列表变量加 _list 后缀，qli88 采纳。
- 专家并行（EP）支持：tjtanaa 询问是否支持 EP，若支持需添加测试；qli88 后来添加了 EP 测试用例。
  - 最终由 tjtanaa 批准（LGTM）。
- 测试代码风格：类风格 vs 函数风格 (style): qli88 移除了类风格测试，重构为纯函数式 pytest 测试。
- 变量命名：添加 _list 后缀 (style): qli88 添加了 _list 后缀，如 shapes_list 等。
- 专家并行（EP）支持 (design): qli88 回复“添加了 EP 测试用例”，后续提交增加了 EP 端到端测试。

风险与影响

- 风险：
  - 精度风险：反量化后再以 BF16 前向 vs 原生 int4 内核存在数值差异，GSM8K 91.2% 可接受，但其他任务可能需要验证。
  - 性能开销：反量化在加载时一次性完成，但权重存储膨胀为 BF16（2x 显存），可能限制 batch size。
  - 兼容性：Int4EmulationTritonExperts._supports_current_device 仅返回 is_cuda_alike()，在非 CUDA/ROCm 平台测试会被跳过。
  - 耦合风险：该实现依赖 TritonExperts 的接口，若基类变更需同步维护。
- 影响：
  - 用户：ROCm 用户可通过 --moe-backend emulation 运行 int4 量化 MoE 模型（如 MiniMax-M3-AWQ-INT4），此前不可用。
  - 系统：新增一个后端选择分支，不影响现有 Marlin/Humming 等后端，但压缩张量方法中 emulation 路由暂未与 --moe-backend 完全解耦（见 TODO 注释）。
  - 团队：需要维护新的模拟路径，但代码量小且模型层单一（继承自 TritonExperts），维护成本可控。
  - 风险标记：精度风险，显存翻倍，仅 CUDA-like 设备，新增维护分支，压缩张量路由未完全解耦

关联脉络

- PR #47718 [ROCm][Perf] DSv4 two-stage compressor kernel for HCA prefill: 同为 ROCm 平台优化，但主题不直接相关；都属于 ROCm 生态建设。
- PR #47881 [Feature] Migrate moe sp support to non-torch compiled path for GLM5.2: 涉及 MoE 后端迁移，与本 PR 的 emulation 后端属于同一子系统（MoE）。