PR #39801 完整报告

vllm-project/vllm

[ROCm][CI] Add missing quantization methods and fix online quant test failures

合并时间：2026-04-28 04:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39801

执行摘要

- 一句话：补充 ROCm 量化注册并修复测试
- 推荐动作：该 PR 已合并，值得所有维护 ROCm 后端的工程师精读。核心学习点包括：（1）平台抽象层如何通过 supported_quantization 列表控制量化方法可见性；（2）测试中通过 is_cuda_alike() 而非 is_cuda() 实现多平台兼容的模式；（3）get_current_memory_usage 应使用 max_memory_allocated 而非 total-free 以确保准确性。对于关注 Quark 量化或 MXFP4 MoE 的开发者，quark_moe.py 中的仿真条件设计值得参考。

功能与动机

PR body 指出：Several quantization methods were failing on ROCm because they weren't registered in the platform's supported_quantization list, even though the underlying kernels already support them (either natively or through emulation fallbacks). 同时测试中峰值内存统计被 is_cuda() 门控从而在 ROCm 上未触发，且内存阈值硬编码为 CUDA 的格式，导致 test_online_quant_peak_mem 失败。

实现拆解

实现拆解分为 6 步：

1. 注册缺失量化方法到平台支持列表（vllm/platforms/rocm.py） 在 RocmPlatform.supported_quantization 中添加 fp8_per_tensor、fp8_per_block、online、mxfp8、modelopt、modelopt_mxfp8、modelopt_mixed、gpt_oss_mxfp4（原有顺序调整）。这些方法已有对应的 ROCmFP8ScaledMMLinearKernel 或 EmulationMxfp8LinearKernel（#39205）等 kernel。
2. 修复峰值内存日志与计算方式（vllm/platforms/rocm.py、vllm/model_executor/model_loader/base_loader.py） 在 base_loader.py 中将 is_cuda() 条件改为 is_cuda_alike() 使日志在 ROCm 上输出；在 rocm.py 中将 get_current_memory_usage 从 total - free 改为 torch.cuda.max_memory_allocated(device)（带前缀 empty_cache() 清零）、保持与 CUDA 一致的内存统计语义。
3. 修复 native 路径的 bias 遗漏（vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py） apply_weights 的 gemm_with_dynamic_quant 分支缺少 bias 加法

，现在在外部补加 y = y + bias if bias is not None，使 native 路径行为与 emulate 路径的 F.linear 一致。

4. 严格化 MoE 仿真条件（vllm/model_executor/layers/quantization/quark/quark_moe.py）将原先 not self.ocp_mx_scheme.startswith('w_mxfp4') 替换为 self.ocp_mx_scheme not in ('w_mxfp4',) 常量，明确只有纯 w4 方案允许跳过仿真，混合方案（如 w_mxfp4_a_mxfp6_*）强制走仿真避免不支持的 kernel dispatch。
5. 处理 1-D scaling 张量兼容性（vllm/model_executor/kernels/linear/scaled_mm/pytorch.py） TorchFP8ScaledMMLinearKernel.apply_scaled_mm 中对 Bs 和 As 增加维度检测：若为一维则 reshape 为 (1, n) 或 (-1, 1)，满足 torch._scaled_mm 的 rowwise 契约。
6. 修复测试中 CPU LAPACK 依赖（tests/quantization/test_turboquant.py）generate_rotation_matrix 将 QR 分解移至 GPU 执行，因为 ROCm PyTorch wheel 不带 CPU LAPACK。

配套测试调整：tests/quantization/utils.py 中的内存阈值改为平台无关；tests/quantization/test_configs.py 和 tests/quantization/test_mixed_precision.py 把 is_cuda() 改为 is_cuda_alike()。

关键文件：

- vllm/platforms/rocm.py（模块 平台层；类别 source；类型 core-logic；符号 supported_quantization, get_current_memory_usage）：核心变更：添加 8 种量化方法到 supported_quantization 列表，并重构 get_current_memory_usage 内存计算方法，直接影响 ROCm 上所有量化方法的可用性和峰值内存日志的正确性。
- vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py（模块 量化层；类别 source；类型 data-contract；符号 apply_weights）：修复 native 路径下 gemm_with_dynamic_quant 缺失 bias 的问题，确保与 emulate 路径行为一致，影响所有使用 OCP_MX 方案的线性层（如 qkv_proj）。
- vllm/model_executor/kernels/linear/scaled_mm/pytorch.py（模块 线性层；类别 source；类型 data-contract；符号 TorchFP8ScaledMMLinearKernel.apply_scaled_mm）：修复 torch._scaled_mm 在 1-D scale 张量上的兼容性，使 ModelOpt FP8_PER_CHANNEL_PER_TOKEN 路径可正常工作于 ROCm。
- vllm/model_executor/model_loader/base_loader.py（模块 模型加载；类别 source；类型 data-contract）：将峰值内存日志条件从 is_cuda() 改为 is_cuda_alike()，使 ROCm 平台能记录模型加载后的峰值内存，是测试修复的关键一环。
- tests/quantization/test_turboquant.py（模块 测试；类别 test；类型 test-coverage；符号 generate_rotation_matrix）：修复 ROCm 上缺失 CPU LAPACK 导致的 QR 分解失败，将计算移至 GPU。
- tests/quantization/utils.py（模块 测试工具；类别 test；类型 test-coverage）：调整峰值内存测试的期望值以匹配平台无关计算，配合 get_current_memory_usage 修改。
- tests/quantization/test_configs.py（模块 测试；类别 test；类型 test-coverage）：将 is_cuda() 改为 is_cuda_alike() 以兼容 ROCm。
- tests/quantization/test_mixed_precision.py（模块 测试；类别 test；类型 test-coverage）：将 is_cuda() 改为 is_cuda_alike() 以兼容 ROCm。

- vllm/model_executor/layers/quantization/quark/quark_moe.py（模块 量化层；类别 source；类型 data-contract；符号 init）：修复 MoE 仿真条件，避免不支持的 OCP MX 方案错误地跳过仿真导致 kernel dispatch 失败。

关键符号：apply_weights, init, apply_scaled_mm, get_current_memory_usage, verify_quantization, generate_rotation_matrix

关键源码片段

vllm/platforms/rocm.py

核心变更：添加 8 种量化方法到 supported_quantization 列表，并重构 get_current_memory_usage 内存计算方法，直接影响 ROCm 上所有量化方法的可用性和峰值内存日志的正确性。

```
class RocmPlatform(Platform):
    # ... ( 其他成员 )
    supported_quantization: list[str] = [
        "awq",
        "awq_marlin",
        "gptq",
        "gptq_marlin",
        "fp8",
        "compressed-tensors",
        "fbgemm_fp8",
        "gguf",
        "quark",
        "mxfp4",
        "mxfp8", # 新增：MXFP8 量化
        "torchao",
        "bitsandbytes",
        "modelopt", # 新增：ModelOpt 基础框架
        "modelopt_fp4",
        "modelopt_mxfp8", # 新增：ModelOpt MXFP8
        "modelopt_mixed", # 新增：ModelOpt 混合精度
        "fp8_per_tensor", # 新增：每张量 FP8 缩放
        "fp8_per_block", # 新增：每块 FP8 缩放
        "online", # 新增：在线量化
        "gpt_oss_mxfp4",
    ]

    @classmethod
    def get_current_memory_usage(
        cls, device: torch.types.Device | None = None
    ) -> float:
        # 先清空缓存再重置峰值计数器，确保统计的是模型加载后的峰值
        torch.cuda.empty_cache()
        torch.cuda.reset_peak_memory_stats(device)
        # 使用 max_memory_allocated 返回峰值分配内存，
        # 而非 total - free（后者受 HIP 运行时等后台开销影响）
```

```
        return torch.cuda.max_memory_allocated(device)
```

vllm/model_executor/layers/quantization/quark/schemes/quark_ocp_mx.py

修复 native 路径下 gemm_with_dynamic_quant 缺失 bias 的问题，确保与 emulate 路径行为一致，影响所有使用 OCP_MX 方案的线性层（如 qkv_proj）。

```
def apply_weights(
    self,
    layer: torch.nn.Module,
    x: torch.Tensor,
    bias: torch.Tensor | None = None,
) -> torch.Tensor:
    if self.emulate:
        # 仿真路径：先反量化权重、量化再反量化激活，然后使用 F.linear
        dq_w = self.dequant_func(layer.weight, layer.weight_scale, x.dtype)
        qdq_x = self.quant_dequant_func(x)
        return F.linear(qdq_x, dq_w, bias)
    # native 路径：调用自定义 GEMM 算子（无 bias 参数）
    y = torch.ops.vllm.gemm_with_dynamic_quant(
        x,
        layer.weight,
        layer.weight_scale,
        self.rocm_use_aiter_fp4_asm_gemm,
        self.out_dtype,
    )
    # gemm_with_dynamic_quant 不携带 bias，手动补加使其与
    # F.linear 行为一致（例如 qkv_proj 使用 qkv_bias=True 时）
    if bias is not None:
        y = y + bias
    return y
```

vllm/model_executor/layers/quantization/quark/quark_moe.py

修复 MoE 仿真条件，避免不支持的 OCP MX 方案错误地跳过仿真导致 kernel dispatch 失败。

```
# 在 QuarkOCP_MX_MoEMethod.__init__ 中
# 定义当前被 AITER 原生支持的 OCP MX 方案元组
# TODO(aiter): 待 rocm_aiter_fused_experts 扩展后更新此列表
_AITER_NATIVE_OCP_MX_SCHEMES = ("w_mxfp4",)

self.emulate = (
    not current_platform.supports_mx()
    or self.ocp_mx_scheme not in _AITER_NATIVE_OCP_MX_SCHEMES
) and (
    self.mxfp4_backend is Mxfp4MoeBackend.NONE or not self.use_rocm_aiter_moe
)
# 原先使用 `not self.ocp_mx_scheme.startswith("w_mxfp4")`，
# 会错误地将 w_mxfp4_a_mxfp6_* 等混合方案也判为 native，
# 但 AITER kernel 不支持这些混合方案，会导致 QuantMethod.NO 错误。
# 改用显式元组后只有纯 w4a16 方案能走 native，其余走仿真。
```

评论区精华

代码审查中有一重要讨论：

- gfx950 的 FP8 格式：gshtras 指出 gfx950 uses fp8_e4m3fn, not fp8_e4m3fnuz，且两种格式占用空间相同，若内存有差异应查找真正原因，而非简单调整阈值。
- 作者回应：AndreasKaratzas 承认差异源于 ROCm 与 CUDA 的内存计算方式不同，在将 get_current_memory_usage 实现对齐后内存增量消失，不再需要调整阈值。
  - 该讨论促成了 get_current_memory_usage 方法的改进，间接提高了内存统计的准确性。
  - gfx950 FP8 格式及内存阈值讨论 (correctness): 确认内存差异源于计算方式而非 FP8 格式，通过统一内存计算方法解决。

风险与影响

- 风险：
  1. 新增量化方法未经验证：虽然注册了列表，但部分方法（如 modelopt_mxfp8）可能仅在仿真模式下可用，性能或数值精度未在 PR 中充分验证，可能出现推理结果错误。
  2. bias 添加可能影响 eager 模式：quark_ocp_mx.py 的 bias 加法在已有联合 bias 的模型中工作正常，但若 gemm_with_dynamic_quant 内部已经处理 bias 则会导致重复添加。从当前逻辑看 gemm_with_dynamic_quant 无 bias 参数，因此是安全的，但需注意未来 kernel 变更。
  3. 内存计算改变对其他平台的影响：get_current_memory_usage 改为 max_memory_allocated 对 CUDA 平台同样生效，若 CUDA 那边之前依赖 total - free 的语义（例如计算峰值）可能发生变化，但新定义更符合“峰值分配”的意图。
  4. 仿真 MoE 性能回退：quark_moe.py 的仿真条件严格化后，w_mxfp4 以外的 MX 方案（如 w_mxfp4_a_mxfp6_*）将走仿真，在 ROCm 上性能可能下降超过 90%（如 Rohan138 在评论中担心的），这是故意牺牲性能换取正确性。
- 影响：
  - 用户影响：ROCm 平台用户现在可以使用包括 fp8_per_tensor、modelopt 系列在内的多种量化方法，提升了推理精度和部署灵活性。test_online_quant_peak_mem 在 ROCm 上通过，增强了 CI 可靠性。
  - 系统影响：无外部 API 或配置变更。
  - 团队影响：加强了 ROCm 平台与 CUDA 平台在量化支持上的对齐，后续新增量化方法时需同时更新平台列表。内存统计方式的统一减少了平台差异。
  - 风险标记：核心路径变更，仿真路径性能回退，测试阈值调整

关联脉络

- PR #39205 [ROCm] Introduce EmulationMxfp8LinearKernel: 本 PR 为 ROCm 注册 mxfp8 和 modelopt_mxfp8 量化方法时依赖此 PR 引入的 EmulationMxfp8LinearKernel 作为仿真回退。