PR #43721 完整报告

vllm-project/vllm

[ROCm][Quantization][4/N] refactor quark_moe fp8 w/ oracle

合并时间：2026-06-23 06:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43721

执行摘要

- 一句话：重构 Quark FP8 MoE 使用 oracle 后端并添加 Qwen3 评测
- 推荐动作：本 PR 是 vLLM 量化后端统一化的重要一步，反映了从手写分支到可扩展 oracle 模式的演进趋势。建议量化相关模块的维护者仔细阅读 quark_moe.py 的变更，理解后端选择流程。新增的评测配置也可作为 FP8 量化质量回归的参考。整体变更清晰，风险可控。

功能与动机

原有 QuarkW8A8Fp8MoEMethod 使用手写的 use_marlin 和 rocm_aiter_moe_enabled 分支来选择后端，与项目中其他量化方法（OCP_MX、NVFP4）的 oracle 模式不一致，导致代码重复且难以维护。通过引入 oracle/fp8.py 的统一后端选择逻辑，可以复用现有的 Fp8MoeBackend 和内核构建函数，消除 marlin_utils_fp8、scalar_types 等旧的依赖。同时，为了验证重构后的 FP8 量化质量，为 AMD Qwen3 模型添加了端到端 GSM8K 评测，并修复了 max_tokens 参数未被正确传递的问题。

实现拆解

1. 移除旧依赖，引入 oracle/fp8：在 quark_moe.py 中删除 marlin_utils_fp8、MoEActivation、scalar_types、envs 等导入，添加 oracle/fp8 模块的 Fp8MoeBackend、make_fp8_moe_quant_config、select_fp8_moe_backend 等符号。
2. 后端选择逻辑重构：在 QuarkW8A8Fp8MoEMethod.__init__ 中，根据 per_channel 和 static_input_scales 确定 weight_key 和 activation_key（如 kFp8StaticChannelSym、kFp8DynamicTokenSym），然后调用 select_fp8_moe_backend() 获取 fp8_backend 和 experts_cls，替代原有的 use_marlin 和 rocm_aiter_moe_enabled 属性。
3. 权重处理统一化：在 process_weights_after_loading 中，使用 fp8_backend.make_quant_config() 获取量化配置，然后调用 fp8_backend.create_weights() 或 convert_to_fp8_moe_kernel_format() 处理权重，移除 AITER 的 shuffle 和 Marlin 的 prepare_fp8_moe_layer_for_marlin 分支。
4. 前向传播更新：在 apply 方法中，使用 self.experts_cls 代替原来的分支，并确保 shared_experts 参数正确传入。此外，create_weights 中的 get_fused_moe_quant_config 也被重构为调用 make_fp8_moe_quant_config。
5. 新增测试与配置：创建两个 YAML 配置文件用于 Qwen3-30B-A3B-Thinking-2507 的 FP8 per-tensor 和 PTPC 变体的 GSM8K 评测；并在 test_gsm8k_correctness.py 的

run_gsm8k_eval 函数中增加 max_tokens=eval_config.get('max_tokens', 256) 参数，使配置文件可以控制生成的最大 token 数。最后在 models-mi3xx-fp8-and-mixed.txt 中注册新配置文件名。

关键文件：

- vllm/model_executor/layers/quantization/quark/quark_moe.py（模块 量化层；类别 source；类型 refactor；符号 QuarkW8A8Fp8MoEMethod.init, QuarkW8A8Fp8MoEMethod.process_weights_after_loading, QuarkW8A8Fp8MoEMethod.apply, _setup_kernel）：核心重构文件，删除手写后端分支，引入 oracle 模式，涉及 init、process_weights_after_loading、apply 等方法的改动。
- tests/evals/gsm8k/configs/Qwen3-30B-A3B-Thinking-2507-FP8.yaml（模块 评测配置；类别 test；类型 test-coverage）：新增 Qwen3 FP8 per-tensor 量化模型的 GSM8K 评测配置，用于验证重构后 FP8 量化的正确性。
- tests/evals/gsm8k/configs/Qwen3-30B-A3B-Thinking-2507-PTPC-FP8.yaml（模块 评测配置；类别 test；类型 test-coverage）：新增 Qwen3 FP8 per-channel/per-token 量化模型的 GSM8K 评测配置。
- tests/evals/gsm8k/test_gsm8k_correctness.py（模块 评测脚本；类别 test；类型 test-coverage；符号 run_gsm8k_eval）：在 run_gsm8k_eval 函数中增加 max_tokens 参数传递，使配置文件可以控制生成的最大 token 数。
- tests/evals/gsm8k/configs/models-mi3xx-fp8-and-mixed.txt（模块 模型清单；类别 docs；类型 documentation）：注册新的 Qwen3 FP8 评测配置文件名，确保新配置被评测框架识别。

关键符号：QuarkW8A8Fp8MoEMethod.init, QuarkW8A8Fp8MoEMethod.process_weights_after_loading, QuarkW8A8Fp8MoEMethod.apply, _setup_kernel, get_fused_moe_quant_config, run_gsm8k_eval

关键源码片段

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

核心重构文件，删除手写后端分支，引入 oracle 模式，涉及 init、process_weights_after_loading、apply 等方法的改动。

以下片段展示了 QuarkW8A8Fp8MoEMethod.__init__ 如何通过 oracle 选择 FP8 后端：

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

class QuarkW8A8Fp8MoEMethod(QuarkMoEMethod):
    """Quark FP8 weight - activation quantization for MoE (W8A8) ."""

    def __init__(
        self,
        moe: FusedMoEConfig,
        per_channel: bool,
        static_input_scales: bool,
    ):
```

```
        super().__init__(moe)
        # 根据 per_channel 和 static_input_scales 确定 oracle 的量化键
        if per_channel:
            # 按通道静态量化权重，按 token 动态量化激活
            weight_key = kFp8StaticChannelSym
            activation_key = kFp8DynamicTokenSym
        elif self.static_input_scales:
            # 全张量静态量化（权重和激活都使用静态 scale）
            weight_key = kFp8StaticTensorSym
            activation_key = kFp8StaticTensorSym
        else:
            # 权重静态 tensor scale，激活动态 token scale
            weight_key = kFp8StaticTensorSym
            activation_key = kFp8DynamicTensorSym

        # 通过 oracle 选择 FP8 后端和对应的专家 kernel 类
        self.fp8_backend, self.experts_cls = select_fp8_moe_backend(
            config=moe,
            weight_key=weight_key,
            activation_key=activation_key,
        )
        # 断言后端和 kernel 类都已正确加载
        assert self.fp8_backend is not None, 'FP8 backend selection failed'
        assert self.experts_cls is not None, 'FP8 expert kernel class not found'
```

评论区精华

- 代码简化建议：@bnellnm 指出 FusedMoEMethodBase 已有 moe_kernel 属性，开发者采纳后移除了冗余定义。
- 健壮性增强：@bnellnm 建议对可能为 None 的值添加 assert 检查，并指出 qc 不应为 None，开发者添加了相应的断言。
- 共享专家兼容性：@bnellnm 提醒 shared_experts 参数需要传递给 make_fp8_moe_kernel 的调用，开发者修复了该问题。
- CI 资源考量：@AndreasKaratzas 表示不能为每个变体引入端到端测试，否则 CI 资源不足；开发者移除了 BF16 基础模型评测配置，仅保留两个 FP8 配置。
- FusedMoEMethodBase 已有 moe_kernel 属性 (design): 开发者采纳，移除了冗余定义。
- 添加 assert 防止 None 值 (correctness): 开发者添加了 assert。
- 传参 shared_experts 给 oracle 后端 (correctness): 开发者修正了调用。
- 减少端到端评测数量 (testing): 开发者移除了基础模型（BF16）评测配置，仅保留两个 FP8 配置。

风险与影响

- 风险：技术风险：

- 重构后 QuarkW8A8Fp8MoEMethod 的行为依赖于 oracle/fp8.py 的实现，如果 select_fp8_moe_backend 对某些罕见配置返回 None，则可能导致运行时错误。尽管代码中已添加 assert，但若 assert 未覆盖所有路径，仍存在风险。
- 删除 envs.VLLM_TEST_FORCE_FP8_MARLIN 可能影响依赖此环境变量的测试或用户工作流，需确认是否有外部调用。
- 新的 GSM8K 评测配置执行 1319 题，每次评测可能耗时较长，可能增加 CI 流水线的整体时长。
- 影响：
- 用户：ROCm 平台使用 Quark FP8 MoE 的用户无需调整代码，推理行为应与之前一致，但由于后端选择统一，某些边缘情况下后端可能不同（例如原本走 Marlin 的现在走 oracle 默认后端）。正确性由 GSM8K 评测保障。
- 系统：代码量减少约 100 行，降低了维护负担。oracle/fp8 模块的复用性得到验证，未来新增量化类型可仿照此模式。
- 团队：开发者需要关注 oracle 后端的迁移进度，确保所有量化类型最终统一到 oracle 架构。
- 风险标记：核心路径变更，量化兼容性，CI 资源开销

关联脉络

- PR #42235 [Kernel][Performance] Add FlashInfer cutedsl NVFP4 GEMM backend: 引入了 oracle 后端选择模式，本 PR 对 FP8 量化采用了相同架构。