PR #39121 完整报告

vllm-project/vllm

[ROCm] Use quant_dtype in per_token_quant instead of hardcoded FP8

合并时间：2026-04-30 04:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39121

执行摘要

- 一句话：修复 ROCm per_token_quant 硬编码 FP8 的 bug
- 推荐动作：该 PR 改动简单清晰，值得快速合并。对于关注 ROCm 量化栈的开发者，可关注后续是否真正启用 int8 路径以及是否在 fake 中添加断言。其他开发者可忽略。

功能与动机

PR body 明确指出 _rocm_aiter_per_token_quant_impl 及其 fake 接受 quant_dtype 参数（可为 torch.int8 或 FP8_DTYPE），但输出张量分配时硬编码了 FP8_DTYPE，导致传入 torch.int8 时仍返回 FP8 张量，这是一个正确性缺陷。修复后输出类型与 quant_dtype 一致，为未来启用 int8 量化路径提供正确基础。

实现拆解

1. 定位硬编码：在 vllm/_aiter_ops.py 文件的 _rocm_aiter_per_token_quant_impl 函数中，第 781 行 torch.empty(x.shape, dtype=FP8_DTYPE, ...) 将输出张量固定为 FP8_DTYPE，与函数签名中的 quant_dtype 参数脱节。
2. 修复真实实现：将第 781 行的 dtype=FP8_DTYPE 替换为 dtype=quant_dtype，使输出张量的数据类型与传入的量化类型一致。
3. 修复 fake 实现：在 _rocm_aiter_per_token_quant_fake 函数中，第 801 行同样硬编码了 FP8_DTYPE，一并改为 quant_dtype，确保 fake 路径的输出类型与真实路径同步。
4. 未新增测试：由于当前所有调用者均传入 FP8_DTYPE，且该 PR 仅涉及两行 dtype 参数的调整，作者未添加独立测试，但 review 中讨论过应添加断言来保持一致性。

关键文件：

- vllm/_aiter_ops.py（模块 ROCm 量化；类别 source；类型 core-logic；符号 _rocm_aiter_per_token_quant_impl, _rocm_aiter_per_token_quant_fake）：ROCm 量化操作的实现文件，包含 per_token_quant 的真实和 fake 函数，是本次改动的唯一文件。

关键符号：_rocm_aiter_per_token_quant_impl, _rocm_aiter_per_token_quant_fake

关键源码片段

vllm/_aiter_ops.py

ROCm 量化操作的实现文件，包含 per_token_quant 的真实和 fake 函数，是本次改动的唯一文件。

```
# vllm/_aiter_ops.py ( 片段 )

def _rocm_aiter_per_token_quant_impl(
    x: torch.Tensor, quant_dtype: torch.dtype, scale: torch.Tensor | None = None
) -> tuple[torch.Tensor, torch.Tensor]:
    from aiter.ops.quant import dynamic_per_token_scaled_quant

    assert quant_dtype in [torch.int8, FP8_DTYPE]

    out_shape = x.shape
    # 修复前：dtype=FP8_DTYPE 硬编码；修复后：使用 quant_dtype 参数
    out = torch.empty(x.shape, dtype=quant_dtype, device=x.device)
    if scale is None:
        scale = torch.empty((*out_shape[:-1], 1), dtype=torch.float32, device=x.device)
    dynamic_per_token_scaled_quant(
        out, x, scale,
        scale_ub=None, shuffle_scale=False, num_rows=None, num_rows_factor=1,
    )
    return out, scale


def _rocm_aiter_per_token_quant_fake(
    x: torch.Tensor, quant_dtype: torch.dtype, scale: torch.Tensor | None = None
) -> tuple[torch.Tensor, torch.Tensor]:
    out_shape = x.shape
    # 同理，fake 实现也一并修复，使用 quant_dtype 而非 FP8_DTYPE
    return (
        torch.empty(x.shape, dtype=quant_dtype, device=x.device),
        torch.empty((*out_shape[:-1], 1), dtype=torch.float32, device=x.device),
    )
```

评论区精华

Review 中 gemini-code-assist[bot] 提出在 fake 实现中添加与真实实现相同的 assert quant_dtype in [torch.int8, FP8_DTYPE] 断言，以保持契约一致性，避免在 fake 路径上因非法 dtype 产生静默错误。该建议未被采纳，可能是因为当前调用者固定传入 FP8_DTYPE，且 fake 仅用于测试 / 编译，实际运行时不易触发。另一位 reviewer AndreasKaratzas 明确表示 I think this one is actually correct.，最终维护者 tjtanaa 给出了 LGTM 并合并。

- fake 实现缺少 dtype 断言 (correctness): 未采纳该建议，reviewer AndreasKaratzas 认为当前实现正确，维护者 tjtanaa 批准合并。

风险与影响

- 风险：风险极低。改动仅涉及两个 torch.empty() 调用中的 dtype 参数，将硬编码值替换为函数参数。当前所有调用者均传入 FP8_DTYPE，因此行为无变化。未来如果调用者传入

torch.int8，输出将正确为 int8 类型，不再返回错误的 FP8 张量。fake 实现缺少断言，若将来传入非法 dtype 可能产生静默类型错误，但 fake 路径通常不用于生产推理，风险可控。

- 影响：影响范围极小，仅涉及 ROCm 后端的 per_token_quant 量化路径。对现有用户无影响（所有调用者仍传入 FP8_DTYPE），但为将来支持 int8 量化铺平了道路。无性能退化，无 API 变更。
- 风险标记：缺少测试覆盖

关联脉络

- PR #36092 Fix other _aiter_ops bugs: PR body 提及，该 PR 解决了 _aiter_ops 的其他 bug，但不涉及 per_token_quant 的 dtype 问题，两者互补。