PR #46658 完整报告

vllm-project/vllm

[ROCm][CI] Relax fused layernorm quant test tolerances for one-ULP outliers

合并时间：2026-06-27 10:44

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46658

执行摘要

- 一句话：放松 ROCm 融合层归一化量化测试的 ULP 容忍度
- 推荐动作：值得精读。PR 展示了处理跨平台精度问题的规范方法：分析根因（归约顺序）、定义 ULP 测量工具、在测试中设置平台特定容忍边界，并保持主要路径严格。其提取的 bf16_ulp_distance 和 fp8_ulp_distance 可作为后续 ROCm 测试中的公共组件。

功能与动机

PR body 说明：融合 RMSNorm + quant kernel 与参考路径因 fp32 归约顺序和 FMA 收缩不同，偶尔导致单组 bf16 abs-max 翻转 1 ULP，或元素跨 E4M3 绑定边界，造成 CI 测试失败。这是跨平台精度伪影，而非 kernel 缺陷，因此需要放宽测试容忍度。

实现拆解

1. 提取通用 ULP 距离函数：将 test_fused_deepseek_v4_kv_insert.py 中的私有函数 _bf16_ulp_distance 和 _fp8_ulp_distance 移动到 tests/kernels/utils.py 作为公共函数 bf16_ulp_distance 和 fp8_ulp_distance，并保留相同的 bit 重新解释逻辑。
2. 改进 test_fused_quant_layernorm.py 的 scale 检查：引入内嵌函数 scales_close，在 ROCm + bf16 + 分组量化的条件下，对 scale 比较允许较大的相对容差（rtol=1e-2），而其他路径（per-token、fp32、CUDA）维持原有严格 torch.allclose。
3. fp8 输出异常值容忍：在 test_fused_quant_layernorm.py 中，当 fp8 直接比较失败且为 ROCm 条件时，改用 fp8_ulp_distance 统计异常元素量，允许不超过 numel//100k + 8 个非零 ULP 元素，其余元素必须完全一致。
4. 同步修改 test_layernorm.py：将原先全局的 assert_close(atol/rtol=1e-3) 替换为平台分派：ROCm 使用异常值计数检查，CUDA 保持原样。
5. 清理测试文件：删除 test_fused_deepseek_v4_kv_insert.py 中重复的私有函数定义，改为从 utils 导入。

关键文件：

- tests/kernels/test_fused_deepseek_v4_qnorm_rope_kv_insert.py（模块 Deepseek 内核；类别 test；类型 test-coverage；符号 _bf16_ulp_distance, key, _fp8_ulp_distance）：移除私有 ULP 距离函数，改为导入公共工具，并调整 fp8 精度检查使用 fp8_ulp_distance。
- tests/kernels/core/test_fused_quant_layernorm.py（模块 层归一化；类别 test；类型 test-coverage；符号 scales_close）：引入 scales_close 函数，为 ROCm 上 bf16 分组量

化 scale 和 fp8 输出增加平台特定容忍逻辑。

- tests/kernels/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 bf16_ulp_distance, key, fp8_ulp_distance）：新增 bf16_ulp_distance 和 fp8_ulp_distance 公共函数，供多个测试文件复用，是本次重构的核心。
- tests/kernels/core/test_layernorm.py（模块 层归一化；类别 test；类型 test-coverage）：在 test_fused_rms_norm_quant 中为 ROCm 添加 fp8 异常值容忍检查，CUDA 保持原样。

关键符号：bf16_ulp_distance, fp8_ulp_distance, scales_close

关键源码片段

tests/kernels/core/test_fused_quant_layernorm.py

引入 scales_close 函数，为 ROCm 上 bf16 分组量化 scale 和 fp8 输出增加平台特定容忍逻辑。

```
    # Per-block bf16 scales: allow a small relative tolerance for a few groups
    # whose abs-max flips by one ULP between the fused and reference paths. The
    # per-token and fp32 paths stay strict.
    relax_block_rocm = (
        group_size is not None
        and dtype == torch.bfloat16
        and current_platform.is_rocm()
    )

    def scales_close(rtol: float, atol: float) -> bool:
        if torch.allclose(ref_scales, ops_scales, rtol=rtol, atol=atol):
            return True
        # ROCm bf16 block scales: relax relative tolerance to 1e-2 (~one bf16 ULP)
        return relax_block_rocm and torch.allclose(
            ref_scales, ops_scales, rtol=1e-2, atol=atol
        )

    if quant_dtype == torch.int8:
        assert scales_close(rtol=1e-5, atol=1e-6)
        assert torch.allclose(ref_out, ops_out, atol=1)
    else:
        assert scales_close(rtol=1e-5, atol=1e-8)
        a = ref_out.to(dtype=torch.float32)
        b = ops_out.to(dtype=torch.float32)
        ok = torch.allclose(a, b, atol=1e-6)
        if not ok:
            if relax_block_rocm:
                # ULP-flipped group scale can cross an E4M3 tie; tolerate a
                # bounded count of isolated fp8 outliers.
                ulp = fp8_ulp_distance(ref_out, ops_out)
                max_outliers = ulp.numel() // 100_000 + 8
                ok = int((ulp > 0).sum().item()) <= max_outliers
            else:
```

```
            # CUDA (& non-bf16): compare dequantized values with relaxed tolerance.
            if group_size is None:
                a_deq = a * ref_scales.view(-1, 1)
                b_deq = b * ops_scales.view(-1, 1)
            else:
                a_deq = a * ref_scales.repeat_interleave(group_size[1], dim=1)
                b_deq = b * ops_scales.repeat_interleave(group_size[1], dim=1)
            ok = torch.allclose(a_deq, b_deq, rtol=5e-2, atol=5e-2)
        assert ok
```

tests/kernels/utils.py

新增 bf16_ulp_distance 和 fp8_ulp_distance 公共函数，供多个测试文件复用，是本次重构的核心。

```
def bf16_ulp_distance(a: torch.Tensor, b: torch.Tensor) -> torch.Tensor:
    """返回两个 bf16 张量之间的可表示步长距离。

    通过 reinterpret 为 int16 并应用 IEEE-754 全序映射，
    使得相邻可表示值的距离为 1。
    """

    def key(t: torch.Tensor) -> torch.Tensor:
        # 提取低 16 位并转换为 int64
        u = t.contiguous().view(torch.int16).to(torch.int64) & 0xFFFF
        # 全序映射：负数（MSB 为 1）翻转所有位，正数保持 + 0x8000
        return torch.where(u >= 0x8000, 0xFFFF - u, u + 0x8000)

    return (key(a) - key(b)).abs()


def fp8_ulp_distance(a: torch.Tensor, b: torch.Tensor) -> torch.Tensor:
    """返回两个 fp8 张量之间的可表示步长距离。

    输入必须具有相同的 fp8 编码（例如都是 float8_e4m3fn）。
    通过 reinterpret 为 uint8 并应用符号-幅度全序映射。
    """

    def key(t: torch.Tensor) -> torch.Tensor:
        u = t.contiguous().view(torch.uint8).to(torch.int64)
        # 全序映射：负数（MSB=1）翻转所有位，正数保持 + 0x80
        return torch.where(u >= 0x80, 0xFF - u, u + 0x80)

    return (key(a) - key(b)).abs()
```

评论区精华

mawong-amd 建议利用已在 #45681 中添加的 ULP 距离工具，并提到可将其通用化到 tests.kernels.utils。AndreasKaratzas 表示同意。tjtanaa 要求仅对 ROCm 放宽断言，CUDA 保留原样。divakar-amd 采纳了所有建议，重构并添加了 ROCm 守卫。最终 tjtanaa

批准。

- 复用已有的 ULP 距离工具 (design): 已采纳，将私有函数转移到 utils.py 并公开。
- 仅对 ROCm 放宽断言 (correctness): 已添加 current_platform.is_rocm() 守卫，CUDA 路径不变。

风险与影响

- 风险：主要风险是放宽测试容忍度可能掩盖真实 kernel 错误。但 PR 提供了充分分析：差异被确认为 1 ULP 级别，影响极少数元素（如 16k 组中 1 组、3M 元素中 2 个），且 CUDA 路径保留原始严格检查。ROCm 特定的容忍度有明确边界（异常值计数上限），整体风险可控。
- 影响：直接影响 ROCm CI 的四个测试文件，使原本因精度极限差异而失败的 *case* 通过，消除 CI 噪声。不影响 CUDA、Intel GPU、CPU 等其他平台。对用户无功能影响，仅改测试代码。
- 风险标记：测试容忍度放宽，仅限 ROCm 路径

关联脉络

- PR #45681 Add ULP distance utilities for fp8/bf16 testing: 提供了最初的 ULP 距离检查模式，本 PR 将其通用化并复用。