

PR #57112 完整报告

vllm-project/vllm

[ROCm][CI] Fix MLA RoPE fused-kernel tests for TheRock image

合并时间: 2026-09-16 13:57

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/57112>

执行摘要

- 一句话: ROCm MLA RoPE 测试改为按 ULP 分级容差
- 推荐动作: 值得精读, 尤其是从事 ROCm 内核、数值测试或 CI 稳定性的工程师: 这是“不要靠调 atol/rtol 掩盖数值差异”的一个正面范例, 用 FMA 预舍入选择推导出 2 product-ULP 的物理上界, 再用失败比例上限约束整体行为。阅读顺序建议: 先看 PR body 中 Triton 3.7 vs 3.8 的 lowering 对比理解动机, 再看 `assert_rope_close_rocm` 的 Tier 1/Tier 2 双段断言与 NeoX/ 交错两套乘积推导, 最后看 `test_concat_and_cache_mla_rope_fused` 如何用 `slot_mapping` 把 KV cache 的 rope 区段收集回 token 顺序、并把 nope 区段保持严格比对。需要留意的设计决策是 bf16 的 `max_fail_fraction=1e-2` 取值依据 (仅由 `eps` 更粗解释) 以及 fp8 分支未被纳入统一模型; 若后续要推广该断言, 应先确认这两个边界。

功能与动机

PR body 指出: `Core Operations Kernels` 测试组在 TheRock 镜像上失败, 根因是 Triton 3.8 升级——RoPE 中 $x * \cos - y * \sin$ 这类计算, Triton 3.7 的 LLVM 后端降为 `FMA_F16(x, cos, MUL_F16(-y, sin))`, 与 HIP 参考融合内核一致, 而 Triton 3.8 改为 `FMA_F16(-y, sin, MUL_F16(x, cos))`, 两种预舍入选择最多相差 2 个乘积 ULP。作者明确写道: “The current tolerances do not adequately capture the subtleties in this regime, which is partly why the tolerances for this test have been adjusted at least 3 separate times”, 因此目标是 rework and unify 该测试在 ROCm 上半精度 dtype 的容差, 而不是再调一次常数。

实现拆解

1. 变更入口: 在 `tests/kernels/core/test_rotary_embedding_mla_cache_fused.py` 模块级新增 `assert_rope_close_rocm` (约 +82 行), 它是整份 PR 的唯一新增逻辑, 替代此前散落在测试主体里的 ROCm 专用 atol/rtol 常量。
2. 容差核心逻辑 (两段式): Tier 1 用 `get_default_atol/get_default_rtol` 做常规比对; 若有元素失败, Tier 2 要求每个失败元素满足 $|actual - ref| \leq 2 * eps * |product|$, 其中 `product` 取该位置两个乘积 (如 $x * \cos$ 与 $-y * \sin$) 中较大者, 并限制失败元素占比不超过 `max_fail_fraction` (bf16 默认 $1e-2$, 其余 $1e-4$)。为此函数会用 `rope_input` 与 `cos_sin_cache[positions]` 在 FP32 下重算逐元素乘积幅值, 分别处理 NeoX 的 $[x * \cos - y * \sin, y * \cos + x * \sin]$ 布局与交错布局 (`inp[..., 0::2] / inp[..., 1::2]`)。这样放宽只作用于乘积量级所允许的 ULP 范围, 而不是对整张张量放宽常数。

3. 测试主体改造: `test_concat_and_cache_mla_rope_fused` 中新增 `k_pe_orig = k_pe.clone()` 与 `query_orig = query.clone()`, 因为融合内核与参考实现都会原地改写输入, 必须保留原始 RoPE 输入用于 Tier 2 的乘积计算。KV cache 比对分支被重写: fp8 路径保持原有校准容差 (`atol=0.004 if rocm_bf16 else 0.001`、`rtol=0.15 if rocm_neox or rocm_bf16 else 0.1`) ; ROCm 且 dtype 为 half/bfloat16 时, 先用 `block_idx = slot_mapping // block_size`、`block_offset = slot_mapping % block_size` 把 `kv_cache[..., kv_lora_rank:]` 的 rope 区段按 token 顺序收集回来, 再交给 `assert_rope_close_rocm`; 最后对 `[..., :kv_lora_rank]` 的 nope 区段单独做默认 `assert_close`, 把放宽严格限制在 RoPE 部分。query 与 `ref_q_pe` 的比对同样切换到 `assert_rope_close_rocm`。
4. 被移除的旧逻辑: 原来的 `elif rocm_bf16: assert_close(kv_cache, ref_kv_cache, atol=0.02, rtol=1e-3)` 与 `elif rocm_neox: assert_close(..., atol=1e-3, rtol=1e-3)`, 以及 query 上的 `atol=0.04 if rocm_bf16`, 全部由 ULP 模型取代; CUDA 与 fp32 路径仍走默认 `torch.testing.assert_close`, 行为不变。
5. 测试 /CI 配套: 无源码、配置或部署改动; PR body 给出测试计划为“ROCm 10 上 with fix (Buildkite #13002) vs without fix (#13000) 跑 Core Operations Kernels 测试组, ROCm 7.2 走本 PR CI”, 结论是 ROCm 10 修复且 7.2/10 无回归。

关键文件:

- `tests/kernels/core/test_rotary_embedding_mla_cache_fused.py` (模块 旋转位置编码; 类别 test; 类型 test-coverage; 符号 `assert_rope_close_rocm`, `test_concat_and_cache_mla_rope_fused`) : 本 PR 唯一变更文件: 新增 `assert_rope_close_rocm` 两段式容差断言, 并重写 `test_concat_and_cache_mla_rope_fused` 的 ROCm 半精度比对路径, 是修复 TheRock/Triton 3.8 CI 失败的全部实现。

关键符号: `assert_rope_close_rocm`, `test_concat_and_cache_mla_rope_fused`

关键源码片段

`tests/kernels/core/test_rotary_embedding_mla_cache_fused.py`

本 PR 唯一变更文件: 新增 `assert_rope_close_rocm` 两段式容差断言, 并重写 `test_concat_and_cache_mla_rope_fused` 的 ROCm 半精度比对路径, 是修复 TheRock/Triton 3.8 CI 失败的全部实现。

```
# tests/kernels/core/test_rotary_embedding_mla_cache_fused.py
def assert_rope_close_rocm(
    actual: torch.Tensor,
    ref: torch.Tensor,
    rope_input: torch.Tensor, # 融合前的原始 RoPE 输入 (query 或 k_pe), 用于重算乘积量级
    cos_sin_cache: torch.Tensor,
    positions: torch.Tensor,
    is_neox: bool,
    *,
    max_fail_fraction: float | None = None,
    atol: float | None = None,
    rtol: float | None = None,
```

) -> None:

"""两段式断言：主体走默认容差，少量“灾难性抵消”元素按乘积 ULP 放行。

Tier 1: $l_{\text{actual}} - r_{\text{ref}} \leq \text{atol} + \text{rtol} * |r_{\text{ref}}|$, 沿用默认容差。

Tier 2: Tier 1 失败的每个元素必须满足

$l_{\text{actual_e}} - r_{\text{ref_e}} \leq 2 * \text{eps} * |r_{\text{product_e}}|$,

且失败元素占比不得超过 `max_fail_fraction`。

之所以是 2 个乘积 ULP：融合内核与参考实现可能把乘加收缩成
预舍入不同乘积的 FMA，两种合法顺序最多相差 2 个 ULP。

"""

if atol is None:

atol = get_default_atol(ref)

if rtol is None:

rtol = get_default_rtol(ref)

if max_fail_fraction is None:

bf16 的 eps 更粗，1 ULP 偏差更容易越过默认容差，故放宽到 1%

max_fail_fraction = 1e-2 if ref.dtype == torch.bfloat16 else 1e-4

eps = torch.finfo(ref.dtype).eps

逐元素乘积幅值，按 (seq, heads, rot_dim) 布局还原

seq_len, rot_dim = rope_input.shape[0], rope_input.shape[-1]

half = rot_dim // 2

inp = rope_input.reshape(seq_len, -1, rot_dim).to(torch.float32)

cs = cos_sin_cache[positions].to(torch.float32) # (seq_len, 2*half)

cos = cs[:, :half].unsqueeze(1) # 沿 head 维广播

sin = cs[:, half:].unsqueeze(1)

x, y = inp[:, :, :half], inp[:, :, half:]

if is_neox:

out = [x*cos - y*sin, y*cos + x*sin]

prod = torch.cat(

[

torch.maximum((x * cos).abs(), (y * sin).abs()),

torch.maximum((y * cos).abs(), (x * sin).abs()),

],

dim=-1,

)

else:

out[2i] = x[2i]*cos[i] - x[2i+1]*sin[i], 交错布局

xe, xo = inp[:, :, 0::2], inp[:, :, 1::2]

prod = torch.stack(

[

torch.maximum((xe * cos).abs(), (xo * sin).abs()),

torch.maximum((xo * cos).abs(), (xe * sin).abs()),

],

dim=-1,

).flatten(-2)

prod = prod.reshape(rope_input.shape)

diff = (actual.to(torch.float32) - ref.to(torch.float32)).abs()

```

tol = atol + rtol * ref.to(torch.float32).abs()
fail = diff > tol
n_fail = int(fail.sum())
if n_fail == 0:
    return
# Tier 2 前置门槛: 失败元素必须足够稀少, 避免整体性数值回归被放过
assert n_fail <= max_fail_fraction * diff.numel(), (
    f"{n_fail} elements beyond default tolerance "
    f"({n_fail / diff.numel():.2%}) > {max_fail_fraction:.2%}); "
    f"max diff {diff.max().item()}"
)
excess = diff[fail] - 2 * eps * prod[fail]
assert (excess <= 0).all(), (
    f"{int((excess > 0).sum())} cancellation elements exceed 2 product-ULPs; "
    f"worst excess {excess.max().item()}"
)

# tests/kernels/core/test_rotary_embedding_mla_cache_fused.py
def test_concat_and_cache_mla_rope_fused(...):
    ...
    # 保留融合前输入: 融合算子与参考实现都会原地改写张量,
    # Tier 2 需要用它重算逐元素乘积幅值
    k_pe_orig = k_pe.clone()
    query_orig = query.clone()

    ... # 执行 opcheck 与 ops.concat_and_cache_mla_rope_fused

    if kv_cache_dtype == "fp8":
        ... # fp8 保持原有经验校准容差, 未纳入 ULP 模型
        torch.testing.assert_close(
            result_temp,
            expected_temp,
            atol=0.004 if rocm_bf16 else 0.001,
            rtol=0.15 if rocm_neox or rocm_bf16 else 0.1,
        )
    elif current_platform.is_rocm() and dtype in (torch.half, torch.bfloat16):
        # 按 slot_mapping 把 KV cache 的 rope 区段收集回 token 顺序
        block_idx = slot_mapping // block_size
        block_offset = slot_mapping % block_size
        fused_k_rope = kv_cache[block_idx, block_offset, kv_lora_rank:]
        ref_k_rope_gathered = ref_kv_cache[block_idx, block_offset, kv_lora_rank:]
        assert_rope_close_rocm(
            fused_k_rope,
            ref_k_rope_gathered,
            k_pe_orig,
            rope.cos_sin_cache,
            positions,
            is_neox_style,
        )

```

```

# nope 区段是逐字节拷贝，仍按默认容差严格比对，放行只限 RoPE 部分
torch.testing.assert_close(
    kv_cache[..., :kv_lora_rank], ref_kv_cache[..., :kv_lora_rank]
)
else:
    torch.testing.assert_close(kv_cache, ref_kv_cache)

if current_platform.is_rocm() and dtype in (torch.half, torch.bfloat16):
    assert_rope_close_rocm(
        query,
        ref_q_pe,
        query_orig,
        rope.cos_sin_cache,
        positions,
        is_neox_style,
    )
else:
    torch.testing.assert_close(
        query,
        ref_q_pe,
        atol=get_default_atol(query),
        rtol=get_default_rtol(query),
    )

```

评论区精华

本 PR 的 review 侧几乎没有技术交锋：[claude\[bot\]](#) 仅留言说明“This pull request is from a fork — automated review is disabled”，需要维护者手动 [@claude review](#) 才会跑一次性审查；随后 [AndreasKaratzas](#) 直接 APPROVED 并执行合并。issue 评论也只有 CI 触发流程：[AndreasKaratzas](#) 连续两次 `/ci run`，第一次被 bot 拒绝（“This PR is 5 commits behind upstream [main](#)”），要求先 merge/rebase 到最新 main，随后才触发 Buildkite CI #89262。真正的“讨论”全部体现在 PR body 的数值论证里：作者解释了 Triton 3.7 → 3.8 的 FMA 预舍入翻转，并链接了历史上三次容差调整（#51083、#46409、#32408）来说明旧做法的不可持续性。

- 不再手动调 atol/rtol，改用按乘积 ULP 的定量容差模型 (design): 结论是引入 `assert_rope_close_rocm`：主体走默认容差，失败元素逐个受 2 product-ULP 约束，且失败占比受限 (fp16 1e-4、bf16 1e-2)，作者称其为 rework and unify 容差。
- fork PR 的自动 review 被禁用，缺少第三方技术复核 (other): 未有人触发一次性 review，最终仅 [AndreasKaratzas](#) 一人 APPROVED 并合并；数值推导细节未获得独立复核。
- CI 触发流程：分支落后 main 导致首次 `/ci run` 被拒 (other): 以一次 merge main 提交解决，CI 正常触发；不影响测试逻辑本身。

风险与影响

- 风险：

1. 检测强度变化: bf16 允许最多 1% 的元素超出默认容差 (此前 atol=0.02 是对全量元素生效), 若内核出现“低频但系统性”的偏差, 只要单元素幅度在 2 个乘积 ULP 内就可能被放行; Tier 2 的逐元素 ULP 上限是目前唯一的幅度护栏, 缺少对该放行比例的独立校验。
2. 覆盖路径扩大: base 版本中 ROCm + fp16 + 非 NeoX 的 kv_cache 走默认 assert_close, 现在转入两段式断言, 等于对 fp16 也引入 1e-4 的失败比例余量, 属于轻微放松。
3. 测试内重实现内核语义: assert_rope_close_rocm 用 Python/FP32 复刻了 NeoX 与交错两种 RoPE 布局的乘积关系; 若将来内核布局或 cos_sin_cache 排布变化而断言未同步, 可能产生“过宽”的假通过或误报, 二者都不易被察觉。
4. 根因未消除: Triton 3.8 / TheRock 镜像上 AITER Triton rope 与融合内核的 1-2 ULP 数值差异依然存在, 本 PR 只是让它被定量容忍, 属于测试侧缓解。
5. 策略并存: fp8 分支仍使用经验校准的 atol/rtol (含 rtol=0.15), 与新 ULP 模型并存, 未来维护者需要理解两套容差语言的边界。
6. 覆盖面: 只改了一个文件, 其他 ROCm 内核测试 (如 MLA decode、MoE) 若遇同类 Triton 3.8 数值漂移, 仍需各自处理。- 影响: 对用户与线上服务无运行时影响——变更完全位于 tests 目录, 不触碰 vllm/model_executor/layers/rotary_embedding 或 concat_and_cache_mla_rope_fused 算子本身。对 CI 与团队影响明确: TheRock/ROCM 10 镜像上失败多日的 Core Operations Kernels 测试组恢复绿色, ROCM 7.2 亦无回归, 解除了 ROCm 侧持续调容差的维护负担。对后续测试作者, assert_rope_close_rocm 提供了一种可复用的“默认容差 + 乘积 ULP 定向放行”范式, 可迁移到其他存在 FMA 收缩差异的融合算子回归测试中; 同时它把 nope 区段恢复为严格比对, 反而提升了该区域的检测严格度。- 风险标记: 仅测试改动未修根因, 容差按 ULP 定向放宽, bf16 允许 1% 元素超差, 测试内重实现 RoPE 布局语义, fp8 与 ULP 两套容差并存

关联脉络

- PR #51083 历史 RoPE 测试容差调整 PR (标题未在上下文中提供): PR body 明确引用为本测试容差此前被反复调整的来源之一, 说明本次 ULP 模型是对这三次调参的替代。
- PR #46409 历史 RoPE 测试容差调整 PR (标题未在上下文中提供): 同上, body 中列为该测试容差第 2 次调整的先例。
- PR #32408 历史 RoPE 测试容差调整 PR (标题未在上下文中提供): 同上, body 中列为该测试容差第 3 次调整的先例。
- PR #55966 [ROCM][Spec Decode] Add Aiter MLA decode support non-causal draft block: 同为 ROCm AITER MLA 相关的内核 / 测试改动, 涉及 MLA 路径的数值与测试断言, 与本 PR 的 ROCm MLA 测试脉络相近。
- PR #56743 [ROCM][Perf] Optimize DSV4.1 K=512 decode top-k on gfx950: 同一时期 ROCm 内核测试 (tests/kernels 目录) 工作, 反映 ROCm 侧持续的内核验证与容差调整节奏。