PR #29694 完整报告

sgl-project/sglang

[AMD] Fix int8 per-token quant Triton portability + register test for AMD nightly CI

合并时间：2026-07-02 03:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29694

执行摘要

- 一句话：修复 AMD 上 int8 量化 Triton 内核编译错误
- 推荐动作：值得精读。本 PR 展示了如何在不破坏 CUDA 路径的前提下，优雅地解决 Triton 内核在 AMD/ROCm 上的可移植性问题。使用 tl.constexpr 分支消除的模式可复用。

功能与动机

AMD ROCm 平台上的 Triton 编译器不再支持 CUDA 特有的 tl.extra.cuda.libdevice.round（对应 __nv_roundf），导致 int8 per-token 量化内核编译时抛出 RuntimeError。本 PR 目标是修复此可移植性问题，并确保 int8 kernel 测试在 AMD nightly CI 中运行。

实现拆解

1. 修改内核文件 python/sglang/srt/layers/quantization/int8_kernel.py：
   - 新增导入 from triton.language.extra import libdevice 和 is_hip 工具函数。
   - 在 _per_token_quant_int8 内核函数中新增 IS_HIP: tl.constexpr 形参，利用 Triton 的常量编译时分支消除特性。
   - 在 round 调用处，根据 IS_HIP 选择：ROCm 上使用 libdevice.round，CUDA 上保持原有 tl.extra.cuda.libdevice.round，确保 CUDA 路径字节级不变。
   - 在调用处 per_token_quant_int8 中传递 IS_HIP=_is_hip。
2. 注册测试到 AMD nightly CI：在 test/registered/quant/test_int8_kernel.py 中，新增导入 register_amd_ci，并添加 register_amd_ci(est_time=15, suite="nightly-amd-kernel-1-gpu", nightly=True)，同时保留原有的 CUDA CI 注册。

关键文件：

- python/sglang/srt/layers/quantization/int8_kernel.py（模块 量化模块；类别 source；类型 dependency-wiring；符号 _per_token_quant_int8, per_token_quant_int8）：核心修复文件，修改了 Triton JIT 内核 _per_token_quant_int8 的 round 调用，新增 IS_HIP 分支和 libdevice 导入。
- test/registered/quant/test_int8_kernel.py（模块 测试注册；类别 test；类型 test-coverage）：新增 AMD nightly CI 注册，确保此内核在 AMD 上持续被测试。

关键符号：_per_token_quant_int8, per_token_quant_int8

关键源码片段

python/sglang/srt/layers/quantization/int8_kernel.py

核心修复文件，修改了 Triton JIT 内核 _per_token_quant_int8 的 round 调用，新增 IS_HIP 分支和 libdevice 导入。

```
# python/sglang/srt/layers/quantization/int8_kernel.py (partial)
import triton
import triton.language as tl
from triton.language.extra import libdevice # 新增：后端无关的 libdevice 模块

from sglang.srt.utils import get_device_name, is_cuda, is_hip # 新增 is_hip

_is_cuda = is_cuda()
_is_hip = is_hip() # 新增 HIP 检测

@triton.jit
def _per_token_quant_int8(
    x_ptr,
    xq_ptr,
    scale_ptr,
    x_sum_ptr,
    stride_x,
    stride_xq,
    N,
    CAL_SUM: tl.constexpr,
    BLOCK: tl.constexpr,
    IS_HIP: tl.constexpr, # 新增：编译时常量，CUDA 上为 False，ROCm 上为 True
):
    row_id = tl.program_id(0)
    cols = tl.arange(0, BLOCK)
    mask = cols < N

    x = tl.load(x_ptr + row_id * stride_x + cols, mask=mask, other=0.0).to(tl.float32)
    absmax = tl.maximum(tl.max(tl.abs(x)), 1e-10)
    scale_x = absmax / 127
    x_q = x * (127 / absmax)
    # 根据 IS_HIP 选择 round 实现：
    if IS_HIP:
        # ROCm Triton 已放弃 CUDA 专用 `tl.extra.cuda.libdevice.*` 包装（__nv_roundf），
        # 改用后端无关的 libdevice.round。
        x_q = libdevice.round(x_q).to(tl.int8)
    else:
        # CUDA 路径保持字节级不变，使用原有的 CUDA 内部函数。
        x_q = tl.extra.cuda.libdevice.round(x_q).to(tl.int8)

    if CAL_SUM:
        x_sum = tl.sum(x, axis=0)
        tl.store(x_sum_ptr + row_id, x_sum.to(x_sum_ptr.dtype.element_ty))
```

```
    tl.store(xq_ptr + row_id * stride_xq + cols, x_q, mask=mask)
    tl.store(scale_ptr + row_id, scale_x.to(scale_ptr.dtype.element_ty))


def per_token_quant_int8(x, scale_dtype=torch.float32, cal_sum=False):
    # ...（省略前面的参数准备代码）
    _per_token_quant_int8[(M,)](
        x, x_q, scales, x_sum,
        stride_x=x.stride(-2),
        stride_xq=x_q.stride(-2),
        N=N,
        CAL_SUM=cal_sum,
        BLOCK=BLOCK,
        IS_HIP=_is_hip, # 传入全局 HIP 标志
        num_warps=num_warps,
        num_stages=1,
    )
    # ...
```

test/registered/quant/test_int8_kernel.py

新增 AMD nightly CI 注册，确保此内核在 AMD 上持续被测试。

```
# test/registered/quant/test_int8_kernel.py (partial)
from sglang.test.ci.ci_register import register_amd_ci, register_cuda_ci # 新增 register_amd_ci

# 保留原有 CUDA CI 注册（base-b 阶段，1-GPU-small 配置）
register_cuda_ci(est_time=15, stage="base-b", runner_config="1-gpu-small")
# 新增 AMD nightly CI 注册（夜间测试套件，1-GPU 配置）
register_amd_ci(est_time=15, suite="nightly-amd-kernel-1-gpu", nightly=True)
```

评论区精华

仅有两个 approval（BBuf 和 HaiShaw），无 review 评论。说明变更过程非常直接，无设计争议。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更高度局部化：仅修改一个 Triton 内核中一个函数的单行代码，且通过 IS_HIP 编译时常量确保 CUDA 路径完全不变。测试注册是增量添加，不影响现有 CUDA CI。唯一潜在风险是 libdevice.round 的行为在 ROCm 上是否与 CUDA 的 round 完全一致，但 PR 作者已在真实 MI350 硬件上验证数值误差小于 3e-3，且与 PyTorch 参考实现一致。
- 影响：直接影响 AMD GPU（MI350+）用户，使得 int8 MoE 量化路径可以在 ROCm 上正常编译运行。不影响 CUDA、MUSA 等其他后端。测试注册确保 AMD nightly CI 中持续覆盖此功能。
- 风险标记：CUDA 路径未经实际测试（但保持字节不变）

关联脉络

- 暂无明显关联 PR