PR #25286 完整报告

sgl-project/sglang

[Gemma4]: Fix FP8 Triton scale layout

合并时间：2026-05-20 05:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25286

执行摘要

- 一句话：修复 Gemma 4 FP8 Triton scale 布局，支持行向量
- 推荐动作：推荐精读，特别是 _as_column_scale 的防御性设计和与入口断言的配合方式。对于需要兼容多种 scale 布局的 kernel 封装，此模式可复用。

功能与动机

Issue #25227 报告 google/gemma-4-31B-it 在 --quantization fp8 --tp 2 下因断言错误崩溃：scale_b 形状为 (1, 8192) 但 kernel 期望 (scale_b.shape[1] == 1)。PR body 详细追踪到 online FP8 量化路径在 per_token_group_quant_fp8 后对 weight_scale 做了 .t().contiguous()，导致出现 (1, N) 行向量。

实现拆解

1. 新增 scale 标准化函数 _as_column_scale：位于 python/sglang/srt/layers/quantization/fp8_kernel.py，处理输入 scale 的各种布局（标量、1D、2D 列向量、2D 行向量），统一转为列向量 (expected_len,1) 或 (1,1)。对于更高维或异常形状直接返回，靠后续断言拦截。
2. 修改 triton_scaled_mm 入口：将原有的 scale.reshape(-1,1) 两行替换为 _as_column_scale 调用，并在标准化后增加 assert scale_a.dim() == 2 and scale_b.dim() == 2 以捕获非 2D 输入。原有形状断言保持不变。
3. 扩展测试覆盖：在 test/registered/quant/test_triton_scaled_mm.py 中添加 (17, 64, 96) 非方形测试配置，并在每个子测试中验证 scale_b 为行向量 (1,N) 时 triton_scaled_mm 输出与参考一致。

关键文件：

- python/sglang/srt/layers/quantization/fp8_kernel.py（模块 量化层；类别 source；类型 core-logic；符号 _as_column_scale）：核心定位：新增 _as_column_scale 函数并修改 triton_scaled_mm 入口以标准化 scale 布局，直接修复 issue #25227。
- test/registered/quant/test_triton_scaled_mm.py（模块 测试；类别 test；类型 test-coverage）：配套测试：添加非方形用例和行向量 scale 回归测试，覆盖修复场景。

关键符号：_as_column_scale

关键源码片段

python/sglang/srt/layers/quantization/fp8_kernel.py

核心定位：新增 _as_column_scale 函数并修改 triton_scaled_mm 入口以标准化 scale 布局，直接修复 issue #25227。

```
def _as_column_scale(scale: torch.Tensor, expected_len: int) -> torch.Tensor:
    # 标量或 1D 张量：reshape 成列向量 (-1, 1)
    if scale.dim() <= 1:
        return scale.reshape(-1, 1)
    # 对于 2D 张量，分情况处理
    if scale.dim() == 2:
        # 已是列向量 (?, 1)，直接返回
        if scale.shape[1] == 1:
            return scale
        # 行向量 (1, expected_len)：转置为列向量 (expected_len, 1)
        if scale.shape[0] == 1 and scale.shape[1] == expected_len:
            return scale.t()
    # 其他形状（如高维张量）原样返回，由后续断言拦截
    return scale

# 在 triton_scaled_mm 入口调用标准化
scale_a = _as_column_scale(scale_a, M)
scale_b = _as_column_scale(scale_b, N)

# 标准化后确保仍是 2D，且满足列向量形状约定
assert scale_a.dim() == 2 and scale_b.dim() == 2
assert scale_a.dtype == scale_b.dtype and scale_a.is_floating_point()
assert scale_a.shape[1] == 1 and (scale_a.shape[0] == 1 or scale_a.shape[0] == M)
assert scale_b.shape[1] == 1 and (scale_b.shape[0] == 1 or scale_b.shape[0] == N)
```

test/registered/quant/test_triton_scaled_mm.py

配套测试：添加非方形用例和行向量 scale 回归测试，覆盖修复场景。

```
# 在 test_basic_cases 中新增非方形配置
test_configs = [
    (32, 32, 32, torch.int8, torch.float16, False),
    (17, 64, 96, torch.int8, torch.float16, False), # 非方形，避免 M==N==K 的退化
    (64, 64, 64, torch.int8, torch.float16, True),
]
# FP8 可用时也加入非方形
if fp8_supported:
    test_configs.append((17, 64, 96, torch.float8_e4m3fn, torch.float16, False))

# 在每个子测试末尾，验证行向量 scale_b 的回归
scale_b_row = scale_b.t().contiguous() # 构造 (1, N) 行向量
triton_out_row_scale = triton_scaled_mm(
    input, weight, scale_a, scale_b_row, out_dtype, bias
)
torch.testing.assert_close(triton_out_row_scale, ref_out, rtol=rtol, atol=atol)
```

评论区精华

Review 讨论：

- gemini-code-assist[bot] 指出 _as_column_scale 缺少显式 dim 检查，可能被高维张量误触导致 reshape 失败，建议增加 dim() == 2 保护。
- Ratish1 确认修改，额外添加了 dim 检查和入口 2D 断言，防止高维张量进入 kernel 路径。
- BBuf 建议增加非方形回归用例 M=17, K=64, N=96，已在后续提交中补充。
- _as_column_scale 缺少维度检查 (correctness): Ratish1 采纳并添加 dim 检查，同时在 triton_scaled_mm 入口增加 2D 断言。
- 添加非方形回归用例 (testing): Ratish1 在后续提交中添加了该用例和行向量 scale 回归。

风险与影响

- 风险：核心变更仅限 _as_column_scale 函数和 triton_scaled_mm 入口的 scale 预处理，影响范围小。通过额外断言和回归测试降低了未预期形状的风险。性能影响可忽略（仅增加少量 reshape/ 转置操作）。风险点：如果未来有其他调用方传入不满足预期的高维 scale，防御性代码会原样返回并触发断言失败，不会静默通过。
- 影响：
  - 用户：解决了 Gemma-4-31B-it + fp8 + tp=2 的加载错误，使该配置可用。
  - 系统：对 triton_scaled_mm 的 scale 输入更灵活，兼容行向量；不影响非 FP8 路径。
  - 团队：新增 helper 函数可作为同类问题的统一入口。
  - 风险标记：核心路径变更，回归测试覆盖

关联脉络

- 暂无明显关联 PR