PR #38093 完整报告

vllm-project/vllm

[Bugfix] Fix scaled_mm output narrowing for 3D input tensors

合并时间：2026-04-20 16:58

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/38093

执行摘要

- 一句话：修复 FP8 scaled_mm 内核中 3D 输入张量输出缩小错误，避免引擎初始化崩溃。
- 推荐动作：建议工程师精读此 PR，关注 _get_num_tokens 辅助函数的设计，它展示了处理张量形状展平时的通用模式。此外，review 讨论中的 DRY 原则实践值得借鉴，有助于提高代码质量。

功能与动机

在初始化 Phi-3.5-vision-instruct-FP8-dynamic 模型时，引擎崩溃并报错 'shape mismatch'，原因是输出缩小步骤使用了 output_shape[0]（仅批次维度）而不是完整令牌数（批次乘以序列长度）。PR body 中描述了具体崩溃场景和修复需求，即修复 scaled_mm 输出缩小对 3D 输入张量的不正确处理。

实现拆解

1. 导入模块并添加辅助函数：在 vllm/model_executor/kernels/linear/scaled_mm/pytorch.py 文件中添加 import math，并定义 _get_num_tokens(output_shape: list) -> int 函数，计算除最后一个维度外的所有维度乘积。
2. 更新内核变体中的 apply_scaled_mm 方法：在 TorchFP8ScaledMMLinearKernel、RowWiseTorchFP8ScaledMMLinearKernel 和 ChannelWiseTorchFP8ScaledMMLinearKernel 类中，用 num_tokens = _get_num_tokens(output_shape) 替换硬编码的 output_shape[0]，确保输出缩小步骤正确处理任意维度输入。
3. 测试验证：PR body 中提及通过运行初始化脚本和预提交钩子验证修复。虽然没有新增测试文件，但修复基于实际崩溃场景验证，并确保前向传播正确。

关键文件：

- vllm/model_executor/kernels/linear/scaled_mm/pytorch.py（模块 线性内核；类别 source；类型 core-logic；符号 _get_num_tokens, TorchFP8ScaledMMLinearKernel.apply_scaled_mm, RowWiseTorchFP8ScaledMMLinearKernel.apply_scaled_mm, ChannelWiseTorchFP8ScaledMMLinearKernel.apply_scaled_mm）：这是唯一修改的文件，包含所有 FP8 scaled_mm torch 内核的实现，修复了输出形状计算的核心逻辑，直接影响模型初始化和前向传播。

关键符号：_get_num_tokens, TorchFP8ScaledMMLinearKernel.apply_scaled_mm, RowWiseTorchFP8ScaledMMLinearKernel.apply_scaled_mm, ChannelWiseTorchFP8ScaledMMLinearKernel.apply_scaled_mm

关键源码片段

vllm/model_executor/kernels/linear/scaled_mm/pytorch.py

这是唯一修改的文件，包含所有 FP8 scaled_mm torch 内核的实现，修复了输出形状计算的核心逻辑，直接影响模型初始化和前向传播。

```
import math # 新增导入，用于计算乘积

def _get_num_tokens(output_shape: list) -> int:
    """
    计算输出形状中的令牌数。
    torch._scaled_mm 处理 2D 张量，输入张量如果是 3D 会被展平。
    如果 output_shape 是 3D，令牌数是除最后一个维度（隐藏维度）外所有维度的乘积。
    """
    return math.prod(output_shape[:-1])

# 示例：在 TorchFP8ScaledMMLinearKernel 的 apply_scaled_mm 方法中使用
class TorchFP8ScaledMMLinearKernel(FP8ScaledMMLinearKernel):
    def apply_scaled_mm(
        self,
        *,
        A: torch.Tensor,
        B: torch.Tensor,
        out_dtype: torch.dtype,
        As: torch.Tensor,
        Bs: torch.Tensor,
        bias: torch.Tensor | None,
        output_shape: list,
    ) -> torch.Tensor:
        output = torch._scaled_mm(
            A, B, out_dtype=out_dtype, scale_a=As, scale_b=Bs, bias=bias
        )
        # 处理 torch 版本差异，确保输出是单个张量
        if type(output) is tuple and len(output) == 2:
            output = output[0]

        num_tokens = _get_num_tokens(output_shape) # 使用辅助函数获取正确令牌数
        return torch.narrow(output, 0, 0, num_tokens).view(*output_shape) # 正确缩小并重塑形状
```

评论区精华

Review 中，gemini-code-assist[bot] 指出 num_tokens 计算逻辑在三个类中重复，建议提取为模块级辅助函数以遵循 DRY 原则。nemanjaudovic 响应并添加了 _get_num_tokens 函数，tjtanaa 随后批准。讨论焦点是代码可维护性改进，无未解决疑虑。

- 代码重复与辅助函数提取 (design): nemanjaudovic 添加了 _get_num_tokens 函数，解决了重复问题，代码更易维护。

风险与影响

- 风险：风险较低：变更仅限于单个文件中的形状计算逻辑，影响范围明确。潜在风险是如果 output_shape 不符合预期（如少于两个维度），_get_num_tokens 可能抛出错误，但鉴于调用上下文，这种情况应已由基类处理。无性能或安全风险，兼容性良好。
- 影响：对用户：修复了使用 3D 输入张量的 FP8 量化模型（如 Phi-3.5）的初始化崩溃问题，提升稳定性和可用性。对系统：确保 FP8 线性内核在多种输入形状下正常工作，支持更广泛的模型配置和平台（如 ROCm）。对团队：代码更简洁，减少重复逻辑，便于未来维护和扩展。
- 风险标记：核心路径变更

关联脉络

- 暂无明显关联 PR