PR #41135 完整报告

vllm-project/vllm

[Bugfix] fix inductor error for dpsk v4

合并时间：2026-04-29 12:18

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41135

执行摘要

- 一句话：修复 DeepSeek V4 在 Inductor 下的 AssertionError
- 推荐动作：值得阅读，展示了如何通过 custom op wrapper 绕过 Inductor 对 Triton kernel 的限制。对于其他遇到类似 inductor 错误的团队有参考价值。设计模式：使用 direct_register_custom_op 提供 opaque boundary。

功能与动机

DeepSeek V4 在 torch.compile 下因 Inductor 分解 triton kernel 时断言失败，模型无法启动。根本原因是 PyTorch 上游 issue #181735（dynamic shapes 下 decompose_triton_kernel_wrapper_functional 断言失败）。需要绕过 Inductor 的分解逻辑。

实现拆解

1. 导入 direct_register_custom_op 工具函数（来自 vllm.utils.torch_utils）。
2. 将原始 fused_inv_rope_fp8_quant 中的 kernel 启动逻辑提取为独立函数 _fused_inv_rope_fp8_quant_kernel_impl，包含 Triton kernel 调用和缓冲区分配。
3. 注册 custom op torch.ops.vllm.fused_inv_rope_fp8_quant_kernel，绑定实现函数和 fake 实现（用于 Inductor 图追踪）。
4. 修改 fused_inv_rope_fp8_quant 函数，改为调用 custom op 而非直接启动 kernel，从而避免 Inductor 分解 Triton kernel wrapper。
5. 测试验证：使用 lm_eval 运行 gsm8k 和 aime26 benchmark，精度达标（gsm8k 0.9515，aime26 100）。

关键文件：

- vllm/v1/attention/ops/deepseek_v4_ops/fused_inv_rope_fp8_quant.py（模块 DeepSeekV4 算子；类别 source；类型 core-logic；符号 _fused_inv_rope_fp8_quant_kernel_impl, _fused_inv_rope_fp8_quant_kernel_fake）：此文件包含核心修复：将 Triton kernel 启动包装为 custom PyTorch op，避免 Inductor 分解导致的断言失败。添加了 _fused_inv_rope_fp8_quant_kernel_impl 和 _fused_inv_rope_fp8_quant_kernel_fake 符号。

关键符号：_fused_inv_rope_fp8_quant_kernel_impl, _fused_inv_rope_fp8_quant_kernel_fake, fused_inv_rope_fp8_quant

关键源码片段

vllm/v1/attention/ops/deepseek_v4_ops/fused_inv_rope_fp8_quant.py

此文件包含核心修复：将 Triton kernel 启动包装为 custom PyTorch op，避免 Inductor 分解导致的断言失败。添加了 _fused_inv_rope_fp8_quant_kernel_impl 和 _fused_inv_rope_fp8_quant_kernel_fake 符号。

```
# 导入直接注册自定义操作的工具
from vllm.utils.torch_utils import direct_register_custom_op

# fused_inv_rope_fp8_quant 函数的核心修改：调用 custom op 替代直接启动 kernel
# 这样 Inductor 不会尝试分解 Triton kernel wrapper，避免 AssertionError
def fused_inv_rope_fp8_quant(o, positions, cos_sin_cache, ...):
    # ... 前面的计算
    # 通过 custom op 启动 kernel，返回 fp8 和 scale 缓冲区
    fp8_buf, scale_buf = torch.ops.vllm.fused_inv_rope_fp8_quant_kernel(
        o, positions, cos_sin_cache, heads_per_group, quant_group_size,
        chunks_per_head, nope_dim % quant_group_size, rope_dim // 2,
        tma_aligned_scales, fp8_max, tma_aligned_T, num_tokens,
        n_groups, d, scale_inner,
    )
    # 转置以匹配外部期望的形状
    return fp8_buf.transpose(0, 1), scale_buf.transpose(0, 1)

# 实际的 kernel 实现：从原 fused_inv_rope_fp8_quant 中提取
def _fused_inv_rope_fp8_quant_kernel_impl(
    o, positions, cos_sin_cache, heads_per_group, quant_group_size,
    chunks_per_head, rope_start, half_rope, tma_aligned_scales,
    fp8_max, tma_aligned_T, num_tokens, n_groups, d, scale_inner,
) -> tuple[torch.Tensor, torch.Tensor]:
    # 分配输出缓冲区
    fp8_buf = torch.empty(
        (n_groups, num_tokens, d), dtype=torch.float8_e4m3fn, device=o.device
    )
    scale_dtype = torch.int32 if tma_aligned_scales else torch.float32
    scale_buf = torch.empty(
        n_groups * scale_inner * tma_aligned_T,
        dtype=scale_dtype, device=o.device,
    ).as_strided(
        (n_groups, num_tokens, scale_inner),
        (scale_inner * tma_aligned_T, 1, tma_aligned_T),
    )
    # 启动 Triton kernel
    grid = (tma_aligned_T, n_groups * heads_per_group)
    _fused_inv_rope_fp8_quant_per_head[grid](
        o, positions, cos_sin_cache, fp8_buf, scale_buf,
        num_tokens, heads_per_group=heads_per_group,
        o_stride_token=o.stride(0), o_stride_head=o.stride(1),
```

```
        eps=1e-10, QUANT_GROUP_SIZE=quant_group_size,
        CHUNKS_PER_HEAD=chunks_per_head,
        ROPE_START=rope_start, HALF_ROPE=half_rope,
        TMA_ALIGNED_SCALES=tma_aligned_scales,
        num_stages=1, launch_pdl=False, num_warps=1,
    )
    return fp8_buf, scale_buf

# Fake 实现也通过 direct_register_custom_op 注册，返回相同形状的空张量
# 以便 Inductor 在图追踪时获得输出元信息，避免运行时形状不匹配
```

评论区精华

无实质性讨论。zou3519 直接 approve。Claude bot 和 Gemini bot 自动评论。

- zou3519 批准 (other): PR 已批准

风险与影响

- 风险： 1) 自定义 op 依赖 PyTorch 的 _custom_op / torch.library API，可能随版本变化。 2) 若上游修复 issue #181735，可能需要回退以保持一致性。 3) Fake 实现必须与真实输出 shape 完全一致，否则导致静默错误。 4) 仅影响 DeepSeek V4 的 torch.compile 路径，其他模型不受影响。
  - 影响：仅影响 DeepSeek V4 模型在 torch.compile 模式（--compilation-config '{"cudagraph_mode":"FULL_AND_PIECEWISE"}'）下的推理。修复后该配置可正常启动，精度与性能正常。不影响其他模型或默认 eager 模式。
  - 风险标记：依赖 PyTorch internal API, 未来 PyTorch 版本可能破坏

关联脉络

- 暂无明显关联 PR