PR #43647 完整报告

vllm-project/vllm

[ROCm][CI] Fix ROCm multimodal Qwen2.5-VL activation compile and Phi4MM ragged image mask handling

合并时间：2026-05-27 10:53

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43647

执行摘要

- 一句话：修复 ROCm 多模态模型编译与 Phi4MM 图片 mask 处理
- 推荐动作：值得精读：理解 ROCm 上绕过原生编译的技巧以及变长张量填充的通用模式（stack_with_pad）。关注 get_act_and_mul_fn 的设计扩展性。

功能与动机

PR body 提到：Fix two ROCm multimodal regressions seen in Buildkite build 68098:

- Avoid compiling the disabled native SiluAndMul path inside the Qwen2.5-VL vision MLP on ROCm, which avoids the ROCm CI torch._subclasses.schema_check_mode import failure during ViT cudagraph warmup while preserving the CUDA path. - Allow Phi4MM image inputs with variable crop counts by accepting list-valued image_attention_mask tensors and padding ragged image tensors before the HF vision encoder.

实现拆解

变更包含三个文件的修改：

1. 激活函数层 (vllm/model_executor/layers/activation.py)：修改 get_act_and_mul_fn 函数签名，新增 compile_native 关键字参数（默认为 True）。当 compile_native=False 且激活函数为 silu 或 swish 时，直接返回 SiluAndMul(compile_native=False)，跳过原生编译路径，从而避免 ROCm 上的编译错误。错误处理也改为 try-except 形式。
2. Qwen2.5-VL 模型 (vllm/model_executor/models/qwen2_5_vl.py)：导入 current_platform，并在视觉编码器的 blocks 初始化时，调用 get_act_and_mul_fn 时传入 compile_native=not current_platform.is_rocm()，即仅在非 ROCm 平台启用原生编译。
3. Phi4MM 模型 (vllm/model_executor/models/phi4mm.py)：新增 stack_with_pad 函数，用于将形状不完全相同的张量列表填充后堆叠为单个批张量；修改 Phi4MMImagePixelInputs 中 image_attention_mask 的类型注解从 torch.Tensor 扩展为 torch.Tensor | list[torch.Tensor]，并添加 dynamic_dims={'nc'}；在 _process_image_input 中对 pixel_values 和 image_attention_mask 应用 stack_with_pad 以处理变长输入。

关键文件：

- vllm/model_executor/models/phi4mm.py（模块 模型；类别 source；类型 data-contract；符号 stack_with_pad）：新增 stack_with_pad 函数和 Phi4MMImagePixelInputs 类型扩展，支持变长 image_attention_mask 输入
- vllm/model_executor/layers/activation.py（模块 激活函数；类别 source；类型 data-contract；符号 get_act_and_mul_fn）：修改 get_act_and_mul_fn 函数，新增 compile_native 参数以绕过 ROCm 上的原生编译
- vllm/model_executor/models/qwen2_5_vl.py（模块 模型；类别 source；类型 data-contract）：在视觉编码器初始化时根据平台决定是否启用原生 SiluAndMul 编译

关键符号：get_act_and_mul_fn, stack_with_pad

关键源码片段

vllm/model_executor/models/phi4mm.py

新增 stack_with_pad 函数和 Phi4MMImagePixelInputs 类型扩展，支持变长 image_attention_mask 输入

```
# vllm/model_executor/models/phi4mm.py

def stack_with_pad(
    tensors: torch.Tensor | list[torch.Tensor],
    padding_value: int | float = 0,
) -> torch.Tensor:
    """
    Stack tensors, padding dimensions that differ across items.

    如果输入已经是单个 Tensor，直接返回；若所有张量形状相同，
    则使用 torch.stack 避免 Python 循环开销。否则创建填充
    后的输出张量，逐个赋值。
    """
    if isinstance(tensors, torch.Tensor):
        return tensors

    assert len(tensors) > 0, "Cannot stack an empty tensor list"
    first_shape = tensors[0].shape
    if all(t.shape == first_shape for t in tensors):
        return torch.stack(tensors)

    ndim = tensors[0].dim()
    assert all(t.dim() == ndim for t in tensors[1:]), (
        "All tensors must have the same number of dimensions"
    )

    out_size = [
        len(tensors),
        *(max(t.shape[i] for t in tensors) for i in range(ndim)),
    ]
    output = tensors[0].new_full(out_size, padding_value)
```

```
    for idx, tensor in enumerate(tensors):
        slices = [idx, *(slice(0, size) for size in tensor.shape)]
        output[tuple(slices)] = tensor

    return output
```

vllm/model_executor/layers/activation.py

修改 get_act_and_mul_fn 函数，新增 compile_native 参数以绕过 ROCm 上的原生编译

```
# vllm/model_executor/layers/activation.py

def get_act_and_mul_fn(act_fn_name: str, *, compile_native: bool = True) -> nn.Module:
    """Get an activation-and-mul (i.e. SiluAndMul) function by name."""
    act_fn_name = act_fn_name.lower()

    # ROCm 上禁用原生编译，避免 cudagraph 预热时导入失败
    if not compile_native and act_fn_name in ("silu", "swish"):
        return SiluAndMul(compile_native=False)

    try:
        return _ACTIVATION_AND_MUL_REGISTRY[act_fn_name]
    except KeyError:
        raise ValueError(
            f"Activation function {act_fn_name!r} is not supported."
        ) from None
```

评论区精华

在 review 中，gemini-code-assist[bot] 建议优化 stack_with_pad：当所有输入张量形状相同时直接使用 torch.stack 以避免 Python 循环的开销。作者 AndreasKaratzas 回复 “Addressed.”，在最终的 commit 中已加入该优化分支（if all(t.shape == first_shape for t in tensors): return torch.stack(tensors)）。该建议被采纳并体现在代码中。

- stack_with_pad 性能优化 (performance): 作者采纳建议，在 stack_with_pad 中添加了形状相同时的快速路径

风险与影响

- 风险：低风险：变更集中在两个模型（Qwen2.5-VL 和 Phi4MM）上，且仅影响 ROCm 平台。get_act_and_mul_fn 的默认行为不变（compile_native=True），对 CUDA 无影响。stack_with_pad 是一个新函数，仅在 Phi4MM 中使用，逻辑简单且包含边界断言。未涉及测试文件变更，但改动较小，回归风险低。
- 影响：影响范围限于 ROCm 用户：修复了 Qwen2.5-VL 在 ROCm 上因激活函数编译失败导致的 ViT cudagraph 崩溃，以及 Phi4MM 多裁剪图片输入时 image_attention_mask 形状不匹配的问题。对 CUDA 用户无功能影响，对系统性能无显著影响。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR