PR #24411 完整报告

sgl-project/sglang

[diffusion] Fuse LTX2 split rotary embedding

合并时间：2026-05-05 16:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24411

执行摘要

- 一句话：新增 LTX2 融合 Triton 内核，BF16 路径性能提升约 1%
- 推荐动作：值得阅读，尤其是学习如何在现有 PyTorch 操作中安全插入融合内核的策略。该 PR 展示了保证数值一致性的方法（匹配 BF16 舍入顺序）和条件回退设计，可作为后续类似优化的参考。

功能与动机

LTX2 模型的 DiT 路径中使用了一种特殊的拆分旋转嵌入（split rotary embedding）布局，之前没有针对该布局的融合内核，使用 PyTorch 多次 reshape/broadcast/elementwise 操作存在性能开销。本 PR 通过添加专用 Triton 内核，在保持数值一致性的前提下提升性能。

实现拆解

1. 新增 Triton 内核：在 python/sglang/jit_kernel/diffusion/triton/ltx2_rotary.py 中实现 _ltx2_split_rotary_kernel。该内核采用 2D 网格（[batch*seq_len, num_heads]），内部按照 x 被拆分为两个半头的布局加载 x_first 和 x_second，加载对应的 cos 和 sin，计算旋转公式时严格匹配原始 PyTorch 中 x*cos 先转为 BF16 再参与 addcmul_ 的顺序，保证位精确。
2. 包装函数：编写 apply_ltx2_split_rotary_emb，进行形状校验后分配输出张量，设置 BLOCK_HALF 为 triton.next_power_of_2(half_dim)，启动内核（num_warps=1）。
3. 路由条件：在 python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py 的 apply_split_rotary_emb 函数入口插入条件判断：当 x 为 3D、cos/sin 为 4D、数据类型均为 BF16，且所有张量在 CUDA 上且连续时，转到新内核；否则回退到原有的 PyTorch 实现。导入语句放在条件内以避免全局 import 开销。
4. 测试与验证：未新增显式测试文件，但通过远程内核检查验证了两种目标形状的位精确性（max_abs=0.0），且端到端 benchmark 显示了可测量的性能提升。

关键文件：

- python/sglang/jit_kernel/diffusion/triton/ltx2_rotary.py（模块 JIT 内核；类别 source；类型 dependency-wiring；符号 _ltx2_split_rotary_kernel, apply_ltx2_split_rotary_emb）：本 PR 核心新增的 Triton 融合内核文件，实现了 LTX2 拆分旋转嵌入的高效计算，包含内核函数和包装入口，是性能提升的关键。

- python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py（模块 模型路由；类别 source；类型 data-contract；符号 apply_split_rotary_emb）：修改现有 apply_split_rotary_emb 函数，添加条件路由到新内核。这是将融合内核接入模型推理的关键入口。

关键符号：_ltx2_split_rotary_kernel, apply_ltx2_split_rotary_emb, apply_split_rotary_emb

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/ltx_2.py

修改现有 apply_split_rotary_emb 函数，添加条件路由到新内核。这是将融合内核接入模型推理的关键入口。

```
def apply_split_rotary_emb(
    x: torch.Tensor, freqs: Tuple[torch.Tensor, torch.Tensor]
) -> torch.Tensor:
    cos, sin = freqs

    # 当输入满足特定条件时，使用 LTX2 专用融合内核
    if (
        x.ndim == 3
        and cos.ndim == 4
        and sin.ndim == 4
        and x.dtype == torch.bfloat16
        and cos.dtype == torch.bfloat16
        and sin.dtype == torch.bfloat16
        and x.is_cuda
        and x.is_contiguous()
        and cos.is_cuda
        and sin.is_cuda
    ):
        # 延迟导入以避免全局依赖
        from sglang.jit_kernel.diffusion.triton.ltx2_rotary import (
            apply_ltx2_split_rotary_emb,
        )

        return apply_ltx2_split_rotary_emb(x, cos, sin)

    # 原有 PyTorch 实现（作为 fallback）
    x_dtype = x.dtype
    needs_reshape = False
    if x.ndim != 4 and cos.ndim == 4:
        b = x.shape[0]
        _, h, t, _ = cos.shape
        x = x.reshape(b, t, h, -1).swapaxes(1, 2)
        needs_reshape = True

    last = x.shape[-1]
    if last % 2 != 0:
```

```
        raise ValueError(
            f"Expected x.shape[-1] to be even for split rotary, got {last}."
        )
    r = last // 2

    split_x = x.reshape(*x.shape[:-1], 2, r)
    first_x = split_x[..., :1, :]
    second_x = split_x[..., 1:, :]

    cos_u = cos.unsqueeze(-2)
    sin_u = sin.unsqueeze(-2)

    out = split_x * cos_u
    first_out = out[..., :1, :]
    second_out = out[..., 1:, :]
    first_out.addcmul_(-sin_u, second_x)
    second_out.addcmul_(sin_u, first_x)

    out = out.reshape(*out.shape[:-2], last)
    if needs_reshape:
        out = out.swapaxes(1, 2).reshape(b, t, -1)

    return out
```

评论区精华

PR 未触发人工 review 评论，仅包含自动化 bot 的配额提醒和作者触发的 CI 重新运行指令。无争议或设计讨论。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。新增内核仅替换特定形状和精度下的计算，其余路径完全不变；数值上通过位精确验证；性能提升幅度小且只影响 LTX2 推理阶段。主要风险是新内核在极端形状或非标准 head_dim 下可能未充分测试，但由于条件路由严格，不会错误影响其他情况。
- 影响：对用户：LTX2 模型推理性能轻微提升（约 1%），无功能或接口变化。对系统：新增一个约 90 行的内核文件，对现有代码影响仅限于一个函数内的条件分支。对团队：需要维护新增的 Triton 内核。
- 风险标记：缺少测试覆盖，精度对齐要求

关联脉络

- 暂无明显关联 PR