PR #24798 完整报告

sgl-project/sglang

[Diffusion][NPU][GPU] Fix SANA model execution error

合并时间：2026-05-11 13:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24798

执行摘要

- 一句话：修复 SANA 模型在 NPU 和 GPU 上的执行错误
- 推荐动作：建议精读，尤其是 Gemma2 注意力掩码的重构（从浮点到布尔）和 DPM scheduler 的参数规范化，这是跨后端的通用改进。GEGLU 融合算子的实现方式可作为其他激活函数 NPU 后端的参考。

功能与动机

根据 PR body，作者在解决 PR #23049 的问题时遇到了 SANA 模型的执行错误。NPU 上出现 'Expected attn_mask dtype to be bool or to match query dtype' 错误，GPU 上存在其他错误。需要修复这些错误以使 SANA 模型在 NPU 和 GPU 上都能正确运行。

实现拆解

1. 为 GeluAndMul 激活函数添加 NPU 后端支持：在 python/sglang/multimodal_gen/runtime/layers/activation.py 中，为 GeluAndMul 类新增 forward_npu 方法，使用 torch_npu.npu_geglu 融合算子，并根据 self.approximate 参数设置 approximate 标志。这解决了 NPU 上缺少 GEGLU 融合实现的问题。
2. 重构 Gemma2 注意力掩码生成，使用布尔掩码替代浮点掩码：在 python/sglang/multimodal_gen/runtime/models/encoders/gemma2.py 中，将注意力掩码从 float32 类型的 0/-inf 掩码改为 bool 类型的 True/False 掩码。使用 torch.tril 直接生成因果掩码，滑动窗口和填充掩码通过布尔运算组合。这从根本上修复了 NPU 上 attn_mask 类型不匹配的错误，并提升了效率。
3. 规范化 DPM solver 调度器的参数传递：在 python/sglang/multimodal_gen/runtime/models/schedulers/scheduling_dpm_solver_multistep.py 中，将 step 方法的可变参数 **kwargs 替换为显式命名参数 generator、variance_noise、return_dict，并透传给内部 _inner.step。这解决了 GPU 上因参数传递不完整导致的错误。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/activation.py（模块 激活层；类别 source；类型 core-logic；符号 forward_npu）：核心修复：为 GeluAndMul 激活函数新增 forward_npu 方法，使用 NPU 融合算子实现 GEGLU，是 NPU 支持 SANA 的关键。
- python/sglang/multimodal_gen/runtime/models/encoders/gemma2.py（模块 文本编码器；类别 source；类型 data-contract）：修复 NPU 上注意力掩码类型不匹配错误的根本：

将浮点掩码改为布尔掩码，简化因果掩码生成，并优化滑动窗口和填充掩码逻辑。

- python/sglang/multimodal_gen/runtime/models/schedulers/scheduling_dpm_solver_multistep.py（模块 调度器；类别 source；类型 data-contract）：修复 GPU 上调度器参数错误：将可变参数改为显式命名参数，确保参数正确传递。

关键符号：forward_npu, forward

关键源码片段

python/sglang/multimodal_gen/runtime/layers/activation.py

核心修复：为 GeluAndMul 激活函数新增 forward_npu 方法，使用 NPU 融合算子实现 GEGLU，是 NPU 支持 SANA 的关键。

```
class GeluAndMul(CustomOp):
    """An activation function for GeGLU.
    The function computes x -> GELU(x[:d]) * x[d:] where d = x.shape[-1] // 2.
    """

    def __init__(self, approximate: str = "none"):
        super().__init__()
        self.approximate = approximate
        if approximate not in ("none", "tanh"):
            raise ValueError(f"Unknown approximate mode: {approximate}")

    def forward_cuda(self, *args, **kwargs) -> Any:
        return self.forward_native(*args, **kwargs)

    # 新增 NPU 后端：使用 Ascend 融合算子 npu_geglu
    def forward_npu(self, x: torch.Tensor) -> torch.Tensor:
        # approximate=1 对应 'tanh' 模式，否则为 0
        y_npu, _ = torch_npu.npu_geglu(
            x,
            dim=-1,
            approximate=1 if self.approximate == "tanh" else 0,
            activate_left=True,
        )
        return y_npu

    def forward_native(self, x: torch.Tensor) -> torch.Tensor:
        """PyTorch-native implementation equivalent to forward()."""
        d = x.shape[-1] // 2
        return F.gelu(x[..., :d], approximate=self.approximate) * x[..., d:]
```

python/sglang/multimodal_gen/runtime/models/encoders/gemma2.py

修复 NPU 上注意力掩码类型不匹配错误的根本：将浮点掩码改为布尔掩码，简化因果掩码生成，并优化滑动窗口和填充掩码逻辑。

```
    # 旧版本：使用 float32 掩码（0/-inf），导致 NPU 上类型不匹配错误
    # attn_mask = torch.zeros(...) -> masked_fill(causal, float("-inf"))
```

```
    # 新版本：使用 bool 掩码，避免类型转换，更高效
    attn_mask = torch.tril(
        torch.ones((seq_len, seq_len), device=hidden_states.device, dtype=torch.bool)
    )
    if self.is_sliding and self.sliding_window is not None:
        idx = torch.arange(seq_len, device=hidden_states.device)
        dist = idx[None, :] - idx[:, None]
        too_far = dist > self.sliding_window
        attn_mask = attn_mask.masked_fill(too_far, False)

    if attention_mask is not None:
        attn_mask = attn_mask[None, None, :, :].expand(
            batch_size, 1, seq_len, seq_len
        )
        # 使用布尔与运算合并填充掩码，替代旧版的 masked_fill
        attn_mask = attn_mask & attention_mask.to(torch.bool)[:, None, None, :]

    attn_kwargs = {
        "attn_mask": attn_mask, # 现在是 bool 类型，与 scaled_dot_product_attention 兼容
        "dropout_p": 0.0,
        "is_causal": False,
        "scale": self.scaling,
    }
```

python/sglang/multimodal_gen/runtime/models/schedulers/scheduling_dpm_solver_multistep.py

修复 GPU 上调度器参数错误：将可变参数改为显式命名参数，确保参数正确传递。

```
    # 旧版本：使用 **kwargs 透传，导致部分参数丢失
    # def step(self, model_output, timestep, sample, **kwargs):
    # return self._inner.step(model_output, timestep, sample, **kwargs)

    # 新版本：显式声明所有参数，确保完整传递
    def step(
        self,
        model_output: torch.Tensor,
        timestep: int,
        sample: torch.Tensor,
        generator: torch.Generator | None = None,
        variance_noise: torch.Tensor | None = None,
        return_dict: bool = True,
    ):
        return self._inner.step(
            model_output,
            timestep,
            sample,
            generator=generator,
            variance_noise=variance_noise,
            return_dict=return_dict,
```

)

评论区精华

在代码审查中，gemini-code-assist[bot] 建议简化 Gemma2 的因果掩码创建，使用 ~torch.triu 单次操作替代 torch.ones + torch.triu + masked_fill 的三步操作。作者采纳建议，并实际使用了更简洁的 torch.tril 实现，表示 'Solved, torch.tril is better.' 该讨论已解决。

- 简化 Gemma2 因果掩码创建 (performance): 作者采纳建议，使用 torch.tril 实现，更简洁高效。

风险与影响

- 风险：Gemma2 注意力掩码变更的风险：从浮点掩码切换到布尔掩码，改变了与下游注意力计算的接口。虽然 scaled_dot_product_attention 支持布尔掩码，但若有自定义注意力变体或未来版本变化，需确保兼容性。DPM scheduler 参数签名变更的风险：将 **kwargs 改为显式参数，可能影响调用方传递额外参数，但 diff 显示原有调用未传递额外参数，风险较低。GEGLU NPU 融合的风险：torch_npu.npu_geglu 的近似模式映射（tanh->1，其他->0）需要与 GeluAndMul 的语义对齐，当前实现正确。
- 影响：用户影响：修复了 SANA 模型在 NPU（Ascend）上的执行错误，用户现在可以在 NPU 上运行 SANA 推理；GPU 上的调度器错误也得到修复。系统影响：Gemma2 编码器的注意力掩码性能提升（减少内存分配和操作），DPM scheduler 的代码可维护性提升。团队影响：代码审查中提出的简化建议被采纳，体现了良好的协作。影响范围限于 SANA 扩散模型及相关组件。
- 风险标记：核心路径变更（注意力掩码接口），缺少测试覆盖

关联脉络

- PR #23049 相关 Issue 的 PR（推断）：PR body 提到在解决 PR #23049 的问题时遇到 SANA 模型错误，本 PR 修复这些错误。
- PR #24660 [diffusion] fix: further align ltx2.3 accuracy with tp: 同为 diffusion 模块的修复 PR，涉及类似的后端兼容性问题。