PR #23572 完整报告

sgl-project/sglang

[Diffusion][NPU][Bugfix] Ascend_fa crashes when sequence parallelism is used.

合并时间：2026-04-24 00:21

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23572

执行摘要

- 一句话：修复 AscendFA 在序列并行时头数参数不正确
- 推荐动作：建议合并。这是一个定位准确、改动量小的 bug 修复，修复了 NPU AscendFA 后端在序列并行下的关键崩溃。review 建议已被采纳，代码清晰。后续可考虑添加单元测试覆盖 sp-degree>1 的场景。

功能与动机

修复 AscendFlashAttention 后端在使用 Ulysses 序列并行（ulysses-degree > 1）时的崩溃。用户给出的复现命令中 sp-degree=4（tp-size=2，num-gpus=8），运行时出现 NPU 函数调用失败错误，原因是 query 的 shape[1,5,25200,128] 与预期 [1,20,25200,128] 不匹配——即传入的 num_heads 参数是用全局头数（20）而非切分后的局部头数（20/4=5）。

实现拆解

1. 删除 __init__中的头数缓存（ascend_fa.py 第 70-71 行）：移除 self.num_heads = num_heads 和 self.num_kv_heads = num_kv_heads or num_heads，因为这些值在调用 forward 时可能因序列并行而不再有效。
2. 在 forward 中动态获取头数（ascend_fa.py 第 80 行）：在 forward 方法开始时，通过 query.shape[2] 和 key.shape[2] 获取当前输入张量的实际头数。由于在 transpose(1,2) 之前，第 2 维（索引 2）正好对应头数维度（BNSD 布局中 N 的位置），此时 query 和 key 已经由上层调度根据 sp-degree 切分，因此取到的就是局部头数。
3. 更新 NPU 调用参数（ascend_fa.py 第 94-95 行）：将原本的 num_heads=self.num_heads 和 num_key_value_heads=self.num_kv_heads 改为使用步骤 2 中动态获取的 num_heads 和 num_key_value_heads 变量，确保传入 NPU API 的实际头数与张量匹配。
4. 配套调整：无测试文件变更，仅修改核心逻辑，共 +3/-4 行。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/backends/ascend_fa.py（模块 注意力后端；类别 source；类型 core-logic；符号 AscendFAImpl.init, AscendFAImpl.forward）：核心变更文件，修复 AscendFA 后端在序列并行下头数参数错误导致的崩溃，从缓存值改为动态获取。

关键符号：AscendFAImpl.init, AscendFAImpl.forward

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/backends/ascend_fa.py

核心变更文件，修复 AscendFA 后端在序列并行下头数参数错误导致的崩溃，从缓存值改为动态获取。

```
# 仅展示修改后的关键部分，省略未修改的 imports 和基类
class AscendFAImpl(AttentionImpl):

    def __init__(
        self,
        num_heads: int,
        head_size: int,
        causal: bool,
        softmax_scale: float,
        num_kv_heads: int | None = None,
        prefix: str = "",
        **extra_impl_args,
    ) -> None:
        self.causal = causal
        self.softmax_scale = softmax_scale
        # 注意：不再缓存 num_heads/num_kv_heads，
        # 因为在序列并行（ulysses）时，
        # 构造时传入的全局头数与 forward 时实际的局部头数不一致。

    def forward(
        self,
        query: torch.Tensor,
        key: torch.Tensor,
        value: torch.Tensor,
        attn_metadata: AttentionMetadata,
        return_softmax_lse: bool = False,
    ) -> torch.Tensor:
        # 在 transpose 之前（BNSD 布局，第 2 维是头数），
        # 从输入张量动态获取实际头数，确保序列并行时参数正确。
        num_heads = query.shape[2]
        num_key_value_heads = key.shape[2]
        mask = None
        if self.causal:
            seq_len = query.shape[1]
            mask = torch.triu(
                torch.ones(seq_len, seq_len, device=query.device), diagonal=1
            ).bool()
        # transpose to bs, heads, seq_len, head_dim
        query = query.transpose(1, 2)
        key = key.transpose(1, 2)
        value = value.transpose(1, 2)
```

```
output, lse = torch.ops.npu.npu_fused_infer_attention_score(
    query,
    key,
    value,
    num_heads=num_heads,
    num_key_value_heads=num_key_value_heads,
    scale=self.softmax_scale,
    input_layout="BNSD",
    softmax_lse_flag=return_softmax_lse,
    atten_mask=mask,
)
output = output.transpose(1, 2)
if return_softmax_lse:
    return output, lse
return output
```

评论区精华

reviewer gemini-code-assist[bot] 指出：在 forward 中 query.shape[1] 同时用于获取 seq_len 和（在 transpose 前的版本中）head counts，两个不同含义使用同一索引可能造成维护混淆。虽然当前实现正确，但建议将 head counts 捕获到局部变量以提高可读性。

后续修改采纳了建议：最终实现中已改为在 forward 开始处（transpose 前）通过 query.shape[2] 和 key.shape[2] 获取头数，与 seq_len（query.shape[1]）分开，解决了 review 中提出的混淆问题。

结论：review 建议被采纳，最终提交中正确分离了 seq_len 和头数的获取。

- 使用 query.shape[1] 同时表示 seq_len 和 head counts 的混淆 (correctness): 最终实现中使用 query.shape[2] 和 key.shape[2] 获取头数，在 transpose 之前操作，避免了与 seq_len 的冲突，提高了可读性和健壮性。

风险与影响

- 风险：
  1. 回归风险低：变更仅影响 AscendFA 后端，且逻辑简单——从缓存值改为动态取值。在非序列并行场景下，query.shape[2] 应与缓存的 num_heads 相同（因为 Ulysses degree=1 时不切分），因此行为不变。
  2. 性能影响：无，仅在 forward 中多了两次 shape 读取，开销可忽略。
  3. 兼容性：仅影响 NPU + AscendFA 后端，不影响其他后端。
  4. 测试覆盖缺失：PR 未添加对应单元测试。考虑到这是针对特定 sp-degree 配置的 bugfix，若后续重构该模块或变更 forward 参数顺序，可能再次引入类似问题。
- 影响：
  1. 用户影响：修复了 NPU 用户在使用 Ulysses 序列并行（如 sp-degree=4，tp-size=2，num-gpus=8）时运行扩散模型（如 Wan2.2-T2V）的崩溃问题，影响面局限在 NPU + AscendFA + 序列并行组合场景。

2. 系统影响：无，代码改动量小（+3/-4 行），不涉及其他模块。
3. 团队影响：对于维护 NPU 后端的开发人员，这是一个清晰的 bug 修复示例，说明序列并行下构造缓存与运行时实际参数可能不一致的风险。 - 风险标记：缺少测试覆盖，核心路径变更

关联脉络

- PR #23198 [diffusion] Fix --warmup-resolutions hang with --enable-cfg-parallel: 同为 diffusion 模块在 NPU 上的 bug 修复，涉及 ascend_fa.py 同一目录的后端，且都是与并行策略相关的修复。