PR #34936 完整报告

sgl-project/sglang

[NPU] [FIX] Fix non-contiguous parameter issue in FIA operator

合并时间：2026-08-20 20:43

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34936

执行摘要

- 一句话：修复 FIA 算子非连续输入导致的 NPU 前向失败
- 推荐动作：值得快速浏览，作为 NPU 算子升级适配的参考案例。重点关注的决策是：在调用侧通过 .contiguous() 显式满足算子新契约，并接受 tensormove 的性能代价；未来可通过在算子内部恢复 AutoContiguous 或在上游保证张量连续来消除该开销。建议补充一个非连续输入场景的单元测试，防止回归。

功能与动机

PR body 明确指出：新版 npu_fused_infer_attention_score 算子对输入参数 k_rope、k_value、v_value 新增了 contiguity 检查，而当前调用传入的是非连续张量，导致算子执行失败，并附上了失败的 CI job 链接（Ascend/sglang actions run）。这是算子版本升级带来的契约变化，必须在调用侧适配。

实现拆解

变更围绕 NPU 后端 FIA 算子调用链路展开，共 3 处修改：

1. 定位问题：在 python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py 的 forward_extend 方法中，k 张量经 split 得到 k_nope 与 k_rope，此类视图通常不连续；v直接透传也可能不连续。新版算子新增的contiguity检查使这些输入直接触发执行失败。
2. 显式调用 .contiguous()：在调用 torch.ops.npu.npu_fused_infer_attention_score 时，将 k_nope、v、k_rope 三个参数分别改为 k_nope.contiguous()、v.contiguous()、k_rope.contiguous()；q_nope 与 q_rope 保持不变。这是对算子升级移除内置 AutoContiguous 行为的适配。
3. 提交与配套：PR 共 6 个 commit，其中 5 个为 Merge branch 'main'，实际逻辑变更仅 1 个 commit。未新增单元测试或文档变更。

该修复位于 attention 前向核心路径，影响所有在 Ascend NPU 上使用 FIA 算子的 extend 请求；修复方式简单直接，但 contiguous() 可能引入张量复制开销。

关键文件：

- python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py（模块 NPU 后端；类别 source；类型 core-logic；符号 forward_extend）：唯一变更文件，位于 NPU 后端 attention 核心路径，修复 FIA 算子非连续输入导致的执行失败。

关键符号：forward_extend

关键源码片段

python/sglang/srt/hardware_backend/npu/attention/ascend_backend.py

唯一变更文件，位于 NPU 后端 attention 核心路径，修复 FIA 算子非连续输入导致的执行失败。

```
# NPU 前向扩展阶段（forward_extend）中调用 FIA 融合注意力算子的关键片段。
# 新版算子升级后移除了内置的 AutoContiguous 行为，要求调用方显式保证
# k_nope、v、k_rope 等参数为连续内存，否则算子执行直接失败。
q_nope, q_rope = q.split([layer.v_head_dim, self.qk_rope_head_dim], dim=-1)
k_nope, k_rope = k.split([layer.v_head_dim, self.qk_rope_head_dim], dim=-1)

# 对非连续输入显式调用 .contiguous()，避免因算子新增 contiguity 检查而报错
attn_output, _ = torch.ops.npu.npu_fused_infer_attention_score(
    q_nope,
    k_nope.contiguous(),
    v.contiguous(),
    query_rope=q_rope,
    key_rope=k_rope.contiguous(),
    num_heads=layer.tp_q_head_num,
    input_layout="TND",
    atten_mask=self.fia_mask,
    sparse_mode=3,
    actual_seq_lengths=self.forward_metadata.seq_lens_list_cumsum,
    actual_seq_lengths_kv=self.forward_metadata.seq_lens_list_cumsum,
    scale=layer.scaling,
    next_tokens=0,
)
```

评论区精华

核心讨论围绕 .contiguous() 的性能代价与必要性展开：

- iforgetmyname 指出 .contiguous() 会引入 tensormove，造成性能退化。
- silencejade 回应：算子旧版本内置了 AutoContiguous，版本升级后该行为被移除，必须由调用方显式处理连续性。
- sglang-npu-bot 曾评论 k_rope.is_contiguoxus() ？（疑为拼写错误），最终该机器人多次 APPROVED。
- 最终 iforgetmyname 也给出 APPROVED，说明性能担忧被接受为必要代价。
- k_rope 连续性检查的写法疑问 (question)：该评论未获得直接文字回复，但后续 sglang-npu-bot 对 PR 给出了 APPROVED，确认改动可接受。
- .contiguous() 引入 tensormove 的性能退化担忧 (performance)：性能代价被接受为必要支出，iforgetmyname 最终 APPROVED，该线程关闭。

风险与影响

- 风险：
  1. 性能退化风险：.contiguous() 在输入非连续时会产生张量拷贝（tensor move），增加显存带宽开销；该路径位于 forward_extend 的每次调用中，对长序列 extend 场景可能有可见影响。
  2. 覆盖不全风险：修改只对 k_nope、v、k_rope 做了 contiguous()，q_nope 与 q_rope 未处理。若未来算子版本对 query 侧也增加连续性检查，将再次出现同类失败。
  3. 测试缺失：PR 未新增针对非连续输入的单元测试，回归防护不足，后续重构可能重新引入该问题。
  4. 兼容性：依赖算子新版本行为，旧版本（内置 AutoContiguous）算子下 contiguous() 属于冗余调用，但语义安全。 - 影响：该修复直接影响 Ascend NPU 后端所有依赖 npu_fused_infer_attention_score 的模型推理路径，属于阻断性 bug 修复：不修复则新版本算子下 NPU 前向直接失败。修复后 NPU 用户可恢复正常推理，但可能伴随微小性能开销。对团队而言，这是一个典型的 " 上游算子契约变化 " 适配案例，提示后续 NPU 算子升级时需要同步检查调用侧的连续性假设。 - 风险标记：核心路径变更，缺少测试覆盖，潜在性能退化

关联脉络

- 暂无明显关联 PR