

PR #39561 完整报告

sgl-project/sglang

[NPU] [bugfix] Fix undefined require_attn_tp_allgather

合并时间: 2026-09-16 11:02

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39561>

执行摘要

- 一句话: 修复 NPU 多 stage PP 启动崩溃, 去掉失效属性引用
- 推荐动作: 值得花几分钟精读, 但不是因为改动的技术难度, 而是因为它揭示了一个可复用的工程教训: 删除公共属性 (#38293) 与新增对该属性的引用 (#38249) 在时间上交错, 最终由 NPU 用户在启动阶段埋单。阅读时建议重点关注三点: ① 作者在 PR body 中的根因回溯链路, 理解「跨 PR 集成缺口」是如何形成的; ② `all_gather_group` 从条件取值改为恒取 `attn_tp_group` 的语义变化, 是否会影响下游 `all-gather` 分支; ③ 该路径为何没有自动化测试, 以及在 NPU CI 中加入 `pp-size > 1` 启动冒烟用例的可行性。若你是 NPU 相关代码的维护者, 建议顺手排查其他平台分支 (XPU/ROCm) 是否存在同样指向 `require_attn_tp_allgather` 的残留引用。

功能与动机

PR body 明确说明: 在 NPU 上以 `--pp-size > 1` 运行时, 调度器启动即崩溃, 报错为 `AttributeError: 'Scheduler' object has no attribute 'require_attn_tp_allgather'`, 位置在 `scheduler_pp_mixin.py` 的 `_pp2_only_send_recv_output_tensors_npu`。作者回溯了根因: 「#38293 has deleted `require_attn_tp_allgather`. The issue was introduced by #38249. Possibly due to a timing gap between #38249 and #38293, the reference to `require_attn_tp_allgather`, which should have been removed, was still kept.」也就是说, 一个删除属性的清理 PR 与一个新增该引用路径的功能 PR 在时间上交错合并, 导致 NPU 的 PP 输出收发路径留下悬空引用, 属于典型的跨 PR 集成缺口。

实现拆解

1. 定位崩溃入口: 问题集中在 `python/sglang/srt/managers/scheduler_pp_mixin.py` 的 `_pp2_only_send_recv_output_tensors_npu`。该函数用一次 `batch_isend_irecv` 同时完成「向下一个 stage 发送输出张量」和「从前一个 stage 接收输出张量」, 是 NPU 上规避 HCCL 上 `isend/irecv` 相互阻塞死锁的关键路径。
2. 修复失效的属性引用: 函数体开头原本通过三目表达式构造 `all_gather_group`——`self.attn_tp_group if self.require_attn_tp_allgather else None`。由于 `require_attn_tp_allgather` 已在 PR#38293 中被删除, 这行代码在 NPU + PP 场景下必然抛出 `AttributeError`。改动把它替换为单行 `all_gather_group = self.attn_tp_group`, 删掉条件判断, 删除 3 行、新增 1 行。

3. 语义层面的变化：修复不只是「去报错」，还改变了取值语义——原先在 `require_attn_tp_allgather` 为假时会向后续通信封装传 `None`（表示不做 attention TP 的 all-gather），现在恒为真实的 `attn_tp_group`。作者在 PR 描述中的口径是「pass `all_gather_group=self.attn_tp_group` unconditionally」，因此这是一次有意的行为统一：NPU PP 路径默认走 attention TP all-gather。下游消费该变量的位置未在提供的材料中展示，是本次评审最需要额外确认的点。
4. 测试与配套：本 PR 没有新增或修改任何测试、文档、配置或部署脚本，仅 1 个源码文件、1 个 commit。验证依赖作者手工跑的两组精度对照：Qwen3-32B + `pp-size=2` 的 debug tensor dumps 比对，以及 Qwen3-VL-4B-Instruct + `pp-size=2` 的多模态并行验证（PR 描述中以截图形式给出）。该路径此前没有自动化用例，因此这类「属性被删后引用残留」的问题缺少 CI 守门。

关键文件：

- `python/sglang/srt/managers/scheduler_pp_mixin.py`（模块 流水并行；类别 source；类型 core-logic；符号 `_pp2_only_send_recv_output_tensors_npu`, `all_gather_group`）：唯一的改动文件。修复点位于 NPU 专用的 `_pp2_only_send_recv_output_tensors_npu`，把对已删除属性 `require_attn_tp_allgather` 的条件引用替换为无条件使用 `self.attn_tp_group`，直接决定了 NPU 上 `--pp-size > 1` 能否启动。
- `python/sglang/srt/managers/scheduler_pp_mixin.py`（模块 流水并行；类别 source；类型 core-logic；符号 `_pp2_only_send_recv_output_tensors_npu`）：同上，补充说明：该文件是整个 PP（pipeline parallel）调度编排的集中地，`_pp2_only_send_recv_output_tensors_npu` 与其它 `_pp2_only_*` 变体共同决定 micro-batch 的收发与 skip 策略，因此这里的行为变更会直接影响 NPU 上的 PP 吞吐与死锁规避逻辑。

关键符号： `_pp2_only_send_recv_output_tensors_npu`

关键源码片段

`python/sglang/srt/managers/scheduler_pp_mixin.py`

唯一的改动文件。修复点位于 NPU 专用的 `_pp2_only_send_recv_output_tensors_npu`，把对已删除属性 `require_attn_tp_allgather` 的条件引用替换为无条件使用 `self.attn_tp_group`，直接决定了 NPU 上 `--pp-size > 1` 能否启动。

```
# python/sglang/srt/managers/scheduler_pp_mixin.py
# NPU 专用的 PP 输出张量收发：用一次 batch_isend_irecv 同时完成「发给下一 stage」
# 与「从前一 stage 接收」，避免 HCCL 上 isend/irecv 相互阻塞导致的死锁。
def _pp2_only_send_recv_output_tensors_npu(
    self: Scheduler,
    next_first_rank_mb_id: int,
    next_mb_id: int,
    mbs: List[ScheduleBatch],
    mb_metadata: List[PPBatchMetadata],
    last_rank_comm_queue: deque[Tuple[torch.Event, PPProxyTensors]],
    pp_outputs: PPProxyTensors | None,
) -> Tuple[
```

```

Optional[PPProxyTensors],
Optional[GenerationBatchResult],
Optional[torch.Event],
List[P2PWork],
]:
    """NPU-specific output tensor send/recv using batch_isend_irecv.

```

Pairs the send of output tensors to the next stage with the recv of output tensors from the previous stage in a single ``batch\_isend\_irecv`` call, avoiding the deadlock that can occur on HCCL when separate isend/irecv calls block waiting for each other.

```

    """

```

```

    next_pp_outputs = None
    d2h_event = None
    batch_result = None
    send_output_work = []

```

```

    # --- 本次修复的核心 (PR#39561) ---
    # 这里原先写作:
    # all_gather_group = (
    # self.attn_tp_group if self.require_attn_tp_allgather else None
    # )
    # 但 require_attn_tp_allgather 属性已在 PR#38293 的清理中被删除,
    # 而引入本函数的 PR#38249 仍保留了对它的引用。两者合入时间交错,
    # 导致 NPU 上以 --pp-size > 1 启动时, 调度器初始化阶段直接抛
    # AttributeError: 'Scheduler' object has no attribute
    # 'require_attn_tp_allgather'。
    # 修复方式: 无条件使用真实的 attention TP 通信组, 去掉条件判断,
    # 既消除属性查找失败, 也把「是否做 attn TP all-gather」的语义统一。
    all_gather_group = self.attn_tp_group

```

```

    # --- Prepare send dict ---
    # NPU 上必须始终发出内容 (完整输出或轻量 skip 标记),
    # 保证对端 batch_isend_irecv 里的 recv 总能匹配到一条 send。
    # 否则 SGLANG_PP_SKIP_PURE_CHUNKED_OUTPUT_COMM 会让相邻 rank 做出不一致的
    # skip 判定 (send 的目标与 recv 的目标是两个不同 micro-batch),
    # 进而死锁, 或迫使使用者彻底关掉该优化 (约 10% 吞吐损失)。
    send_dict: Optional[Dict[str, torch.Tensor]] = None
    if self.pp_group.is_last_rank:
        target_send = mbs[next_first_rank_mb_id]
        if target_send is not None:
            q_event, pp_outputs_to_send = last_rank_comm_queue.popleft()
            if not target_send.forward_mode.is_prebuilt():
                if _pp_can_skip_output_comm(target_send):
                    send_dict = {"__msg_type__": "output", "__skip__": True}
                else:
                    self.device_module.current_stream().wait_event(q_event)
                    send_dict = dict(pp_outputs_to_send.tensors)

```

```

        send_dict["__msg_type__"] = "output"
elif pp_outputs:
    # 中间 rank: 收到上一 stage 的输出后原样转发给下一 stage;
    # __skip__ 标记同样透传, 保证 skip 决策由 last rank 单点决定。
    if pp_outputs.tensors.get("__skip__"):
        send_dict = {"__msg_type__": "output", "__skip__": True}
    else:
        send_dict = dict(pp_outputs.tensors)
        send_dict["__msg_type__"] = "output"

```

评论区精华

本次 PR 几乎没有形成技术讨论: `review_comments_count` 为 0, 唯一的 review 是 NPU 机器人 `sglang-npu-bot` 的 APPROVED (body 为空), 另外机器人还发出了 `/tag-and-rerun-ci` 以触发重跑。PR body 中记录的三个 CI 状态均为失败 (PR Test (Base)、PR Test (Extra)、PR Test (AMD ROCm 10) 都是 X), 但材料里没有针对这些失败项的具体归因或结论, 无法判断是环境抖动还是与本改动相关。真正有价值的信息来自作者在 body 里的根因回溯 (#38249 引入引用、#38293 删除属性、时间差导致残留), 这比 review 环节的讨论更能说明问题性质。

- NPU PP 启动崩溃: 对已删除属性 `require_attn_tp_allgather` 的残留引用 (correctness): 已按作者的方案合入: `net +1/-3`, 条件引用被移除, NPU 上 `--pp-size > 1` 恢复可启动; 但「恒传 `attn_tp_group`」带来的行为差异 (原先可能是 None) 未在讨论中被进一步验证。
- CI 失败状态下的批准与重跑 (testing): 缺少对人类可读的失败归因结论, 无法从现有材料判断失败是否与本改动相关; PR 最终仍被合入 (`merged_by` 为 `sglang-npu-bot`), 重跑结果在材料中不可见。

风险与影响

- 风险: 1) 语义变更未完全闭环 (正确性): 把 `all_gather_group` 从「可能为 None」改为「恒为 `attn_tp_group`」会改变 `_pp2_only_send_recv_output_tensors_npu` 后续 `all-gather` 分支的行为。如果下游封装对 None 与真实 process group 有不同的分支处理, NPU 上可能多做一次 `all-gather` (额外通信开销) 或改变张量形状协商方式。材料中未展示该变量的消费点, 无法从本次上下文确认等价性, 这是最需要补验证的地方。
- 2) 缺少自动化测试 (回归): 仓库里没有 `NPU + pp-size > 1` 的启动冒烟用例, 本次修复靠人工跑 Qwen3-32B / Qwen3-VL-4B-Instruct 截图佐证。同类悬空引用一旦再次出现, 仍只能在用户侧启动崩溃时才被发现。
- 3) 跨 PR 时序耦合 (流程): `require_attn_tp_allgather` 被删除时没有同步清理 NPU 侧新增引用, 说明「删属性」类重构缺少全仓符号引用检查或 `ratchet` 测试兜底。
- 4) CI 可信度 (发布): PR 记录中的三个 PR Test 运行均为失败状态, 而 review 侧只有机器人批准与重跑指令, 存在「以失败状态合入」的观感风险, 需要后续确认重跑是否转绿。
- 5) 影响面收敛: 改动只在 NPU 分支函数内, CUDA / ROCm / CPU 的 PP 路径与 `all_gather_group` 逻辑不受影响, 回归半径小。

- 影响：用户侧：使用 NPU 并开启 `--pp-size > 1` 的用户此前会在调度器初始化阶段直接崩溃、完全无法服务；修复后该配置恢复可用，同时对 Qwen3-32B 这类纯文本大模型与 Qwen3-VL-4B-Instruct 这类多模态模型的 PP 并行都得到验证。

系统侧：影响范围限定在 NPU 的 PP 输出张量收发函数 `_pp2_only_send_recv_output_tensors_npu`，不改变通信顺序、消息协议（仍然使用 `__msg_type__` / `__skip__` 标记）或死锁规避策略，因此对已有非 NPU 部署与非 PP 部署无影响。

团队侧：这是一次典型的「清理 PR 与新功能 PR 交错合入」事故，提示维护者在删除公共属性或方法时，需要以符号级搜索 + 平台分支检查（尤其是 NPU、XPU、ROCm 等旁路代码）为前置动作；同时说明 NPU PP 场景的自动化覆盖仍属空白区。

- 风险标记：语义变更待确认，缺少测试覆盖，跨 PR 时序耦合，NPU 专有路径，CI 状态曾失败

关联脉络

- PR #38293 删除 `require_attn_tp_allgather` 的清理 PR（标题未在提供材料中给出）：PR body 明确指出 `require_attn_tp_allgather` 由 #38293 删除，是本 PR 修复的直接上游：删除动作没有同步清理 NPU PP 路径中的引用，才产生了本次启动崩溃。
- PR #38249 引入 `_pp2_only_send_recv_output_tensors_npu` 的功能 PR（标题未在提供材料中给出）：PR body 说明「The issue was introduced by #38249」，即该 PR 新增了引用 `self.require_attn_tp_allgather` 的代码路径；它与 #38293 的时间差是问题根因。
- PR #38402 `fix(npu): fix hybrid KV transfer with PP prefill in PD disaggregation`：同属 NPU + PP 组合场景的修复线，同样触及 NPU 侧并行通信路径（`disaggregation/conn.py`、`memory_pool.py`）并补了单测，可与本 PR 对照：这类问题在 NPU 上反复出现，但本 PR 未补充测试。