PR #32411 完整报告

sgl-project/sglang

Fix token count localization for replicated attention-TP forwards

合并时间：2026-07-26 07:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32411

执行摘要

- 一句话：修复注意力 TP 复制正向的 token 计数定位
- 推荐动作：该 PR 值得精读，尤其是理解注意力 TP 分片 / 复制模式下的 token 计数定位机制。设计简洁，通过谓词模式提供扩展点，对后续支持更多注意力分发策略具有参考价值。

功能与动机

在注意力 TP 复制正向（replicated forward）场景中，每个 rank 持有完整序列而非分片，但现有的 _attn_tp_local_shard_bounds 始终按分片方式计算局部 token 范围，导致非零 rank 上的 token 计数被截断，丢失真实 token。这会影响依赖局部 token 计数的下游功能（如 logits 处理）。

实现拆解

1. 在 forward_batch_info.py 中新增模块级变量 _attn_tp_sequence_sharded_predicate，类型为 Optional[Callable[[int], bool]]，默认值为 None（表示分片模式）。
2. 新增 register_attn_tp_sequence_sharded_predicate 函数，用于外部集成注入该谓词。
3. 修改 _attn_tp_local_shard_bounds 函数：新增 predicate 检查，若谓词存在且返回 False（即复制模式），则返回完整序列范围（num_tokens_per_dp, 0）；否则按原有分片逻辑计算。
4. 修改导入类型，增加 Callable。
5. 本次变更未包含测试文件，但 PR body 提到焦点回归测试通过。

关键文件：

- python/sglang/srt/model_executor/forward_batch_info.py（模块 模型执行器；类别 source；类型 data-contract；符号 register_attn_tp_sequence_sharded_predicate, _attn_tp_local_shard_bounds）：唯一变更文件；实现了谓词注册与 token 计数定位的核心修复逻辑。

关键符号：register_attn_tp_sequence_sharded_predicate, _attn_tp_local_shard_bounds

关键源码片段

python/sglang/srt/model_executor/forward_batch_info.py

唯一变更文件；实现了谓词注册与 token 计数定位的核心修复逻辑。

```
import sys
from typing import Callable, Optional

# 模块级变量：用于指示当前正向是否为序列分片模式
# None 表示未设置，默认按分片模式处理
_attn_tp_sequence_sharded_predicate: Optional[Callable[[int], bool]] = None


def register_attn_tp_sequence_sharded_predicate(
    predicate: Callable[[int], bool],
) -> None:
    """注册一个谓词函数，用于判断给定 num_tokens_per_dp 下的正向
    是否跨 attn-TP rank 分片（sharded）。如果返回 False，则视为
    复制模式（replicated），每个 rank 拥有完整序列。"""
    global _attn_tp_sequence_sharded_predicate
    _attn_tp_sequence_sharded_predicate = predicate


def _attn_tp_local_shard_bounds(num_tokens_per_dp: int):
    """返回 (tokens_per_rank, rank_offset) 指定该 attn-TP rank 的
    序列切片。复制模式下，每个 rank 获得完整序列（offset 为 0）；
    分片模式下，按 attn_tp_size 均分。"""
    predicate = _attn_tp_sequence_sharded_predicate
    # 如果注册了谓词且返回 False，则为复制模式：每个 rank 持有完整序列
    if predicate is not None and not predicate(num_tokens_per_dp):
        return num_tokens_per_dp, 0
    # 默认（包括未注册谓词）为分片模式
    parallel = get_parallel()
    tokens_per_rank = num_tokens_per_dp // parallel.attn_tp_size
    return tokens_per_rank, tokens_per_rank * parallel.attn_tp_rank
```

评论区精华

PR 作者 merrymercy 自行评论 "approve"，无其他 review 评论。讨论较少，变更经过自行验证后合并。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：新增谓词默认 None，在未注册时行为与之前完全相同（分片模式），因此现有分片正向无回归风险。复制模式依赖外部正确注册谓词，若谓词未注册或注册错误，则仍可能错误分片。
  2. 性能风险：每次 _attn_tp_local_shard_bounds 调用均检查全局变量，开销极低（一次 Python 函数调用），无性能问题。
  3. 兼容性：对外接口仅新增可选注册函数，不影响现有 API。 - 影响：直接影响注意力 TP 复制正向（如某些模型或配置下的全复制注意力），修复 token 计数定位错误。对用户

透明，但需要外部集成（如特定模型后端）主动调用注册函数才能生效。无影响范围外的系统变更。 - 风险标记：缺少测试覆盖

关联脉络

- 暂无明显关联 PR