PR #24851 完整报告

sgl-project/sglang

[Session R3] Add routed_experts_start_len for absolute routing slice control

合并时间：2026-05-11 01:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24851

执行摘要

- 一句话：添加 routed_experts_start_len 参数，支持路由数据绝对切片控制
- 推荐动作：值得精读。该 PR 展示了如何通过一个简单的小参数消除多轮推理场景中的线性瓶颈，设计思路清晰：在数据采集点引入切片控制，从源头减少不必要的 gather 和序列化。对于其他类似的数据收集（如 logprobs、hidden states）也可参考此模式。

功能与动机

多轮 RL rollout 中，return_routed_experts 返回整个对话长度的路由数据，包括已缓存的前缀部分。随对话增长，host gather + ZMQ 负载 O(seqlen) 导致约 1s 的 ITL 峰值，且因 DP 注意力同步机制，单个 rank 的延迟会阻塞整个 batch。PR body 中明确指出这是生产环境 RL 训练循环中的主要 *decode* 瓶颈。

实现拆解

1. 参数定义与传递：在 GenerateReqInput、TokenizedGenerateReqInput、CompletionRequest、ChatCompletionRequest 等数据结构中新增 routed_experts_start_len: int = 0 字段，并在 Engine.generate、Engine.async_generate、TokenizerManager、Scheduler.handle_generate_request 等入口函数中透传。
2. 服务端核心逻辑：在 SchedulerOutputProcessorMixin.maybe_collect_routed_experts 中读取 req.routed_experts_start_len，并将其作为 start_len 参数传递给 BaseTopkCapturer.get_topk()。新增 if not req.return_routed_experts: return 提前退出，避免非可选请求的开销。采集后增加行数校验日志，帮助排查静默错误。
3. 底层数据切片：在 BaseTopkCapturer.get_topk() 中新增 start_len: int = 0 参数，生成索引时从 start_len 开始切片：req_to_token_pool[req_pool_idx][start_len:seqlen-1]，并添加防御性 clamp（min(start_len, seqlen-1)）和非负校验。
4. 参数校验：在 Scheduler.handle_generate_request 中，当 return_routed_experts 为 True 时，校验 routed_experts_start_len 是否在 [0, prompt_tokens] 范围内，越界则 abort 请求并提示。
5. 测试覆盖：新增 TestRoutedExpertsStartLen 测试类，包含 4 个用例：默认行为（等效于 0）、行数正确性、越界 abort、缓存命中场景。测试使用 Qwen3-30B-A3B-FP8 模型，TP=2。

关键文件：

- python/sglang/srt/managers/scheduler_output_processor_mixin.py（模块 调度器；类别 source；类型 core-logic）：核心逻辑变更：maybe_collect_routed_experts 增加切片参数传递、提前退出和行数校验。
- python/sglang/srt/state_capturer/base.py（模块 状态捕获；类别 source；类型 core-logic；符号 start_len）：底层数据采集核心：get_topk 新增 start_len 参数，实现内存索引的切片操作。
- test/registered/rl/test_return_routed_experts.py（模块 测试；类别 test；类型 test-coverage；符号 TestRoutedExpertsStartLen, setUpClass, tearDownClass, _send）：新增测试类 TestRoutedExpertsStartLen，覆盖默认行为、切片正确性、越界终止和缓存命中场景，保障新参数的正确性。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：请求校验和构建：在 handle_generate_request 中增加 start_len 范围校验，越界时直接 abort，并将参数传递给 Req 对象。
- python/sglang/srt/entrypoints/engine.py（模块 引擎；类别 source；类型 core-logic；符号 routed_experts_start_len）：Python API 入口：generate 和 async_generate 均需透传新参数，是外部调用者的直接接口。
- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 core-logic；符号 routed_experts_start_len）：数据模型定义：GenerateReqInput 和 TokenizedGenerateReqInput 新增字段，定义参数默认值和序列化。
- python/sglang/srt/entrypoints/openai/protocol.py（模块 API 协议；类别 source；类型 core-logic；符号 routed_experts_start_len）：REST API 协议：CompletionRequest 和 ChatCompletionRequest 需包含新参数，以支持 OpenAI 兼容接口。
- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic；符号 routed_experts_start_len）：内部请求表示：Req 类新增 routed_experts_start_len 属性，用于在调度器内部传递参数。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 API 路由；类别 source；类型 core-logic）：Chat 服务端点：透传 routed_experts_start_len 参数。
- python/sglang/srt/entrypoints/openai/serving_completions.py（模块 API 路由；类别 source；类型 core-logic）：Completions 服务端点：透传 routed_experts_start_len 参数。
- python/sglang/srt/managers/tokenizer_manager.py（模块 预处理；类别 source；类型 core-logic）：Tokenize 管理器：透传 routed_experts_start_len 参数。
- python/sglang/srt/disaggregation/encode_receiver.py（模块 解聚层；类别 source；类型 core-logic）：解聚编码接收器：透传 routed_experts_start_len 参数，确保解聚场景下参数不丢失。

关键符号：get_topk, maybe_collect_routed_experts, handle_generate_request

关键源码片段

python/sglang/srt/managers/scheduler_output_processor_mixin.py

核心逻辑变更：maybe_collect_routed_experts 增加切片参数传递、提前退出和行数校验。

```
def maybe_collect_routed_experts(self: Scheduler, req: Req):
    """为已完成的请求收集路由专家数据。

    如果请求未设置 return_routed_experts，则立即返回，
    避免非可选请求产生 host gather 开销。

    遵守调用方的绝对起始位置，返回 [start_len, seqlen - 1) 范围的数据。
    默认 start_len 为 0，即返回完整序列。

    如果结果张量的行数与期望值不匹配，会记录软警告，
    以便捕获静默回归。
    """
    # 非选定请求直接返回，避免不必要的开销
    if not req.return_routed_experts:
        return
    capturer = get_global_experts_capturer()
    if capturer is None:
        return
    start_len = req.routed_experts_start_len
    req.routed_experts = capturer.get_topk(
        req_pool_idx=req.req_pool_idx,
        seqlen=req.seqlen,
        req_to_token_pool=self.req_to_token_pool,
        start_len=start_len,
    )

    # 行数校验：期望行数为 seqlen - 1 - start_len，至少为 0
    expected_rows = max(0, req.seqlen - 1 - start_len)
    if (
        req.routed_experts is not None
        and req.routed_experts.shape[0] != expected_rows
    ):
        logger.warning(
            "req %s 的路由专家行数不匹配: 实际 %d, "
            "期望 %d (seqlen=%d, cached_tokens=%d, start_len=%s). "
            "这可能是一个静默错误。",
            req.rid,
            req.routed_experts.shape[0],
            expected_rows,
            req.seqlen,
            req.cached_tokens,
            req.routed_experts_start_len,
        )
```

python/sglang/srt/state_capturer/base.py

底层数据采集核心：get_topk 新增 start_len 参数，实现内存索引的切片操作。

```
def get_topk(
    self,
```

```
    req_pool_idx: int,
    seqlen: int,
    req_to_token_pool: ReqToTokenPool,
    start_len: int = 0, # 新增：绝对起始位置，默认 0 表示从头开始
) -> torch.Tensor:
    # 防御性检查：拒绝负值
    if start_len < 0:
        raise ValueError(f"{start_len=} 必须为非负数")
    # 钳制到合法范围 [0, seqlen-1]，防止 index out of bounds
    start_len = min(start_len, seqlen - 1)
    # 从 start_len 开始切片到最后一个有效位置（seqlen-1）
    # 注意：req_to_token_pool 存储了每个 token 对应的 cache pool 索引
    cache_pool_idx = (
        req_to_token_pool.req_to_token[req_pool_idx][start_len : seqlen - 1]
        .cpu()
        .clone() # 显式 clone 以避免后续修改影响
    )
    return self.host_cache.buffer[cache_pool_idx]
```

评论区精华

Review 过程中，zyzshishui 在 python/sglang/srt/managers/scheduler.py 中建议添加 if recv_req.return_routed_experts and 包裹校验逻辑，确保仅在需要路由专家数据时才进行参数校验。最终 PR 接受了该建议，在提交 cb7575d1639555319f6c5bf3deb770e943ce42d7 中设置了默认值 0 并拒绝负值。另外，Qiaolin-Yu 在最终评论中要求添加文档，PR 在提交 98c7055626d0ff59a9051a8ca734d004c7d049c5 中补充了文档。

- 校验逻辑位置优化 (design)：已采纳，后续提交中添加了外围条件判断。
- 文档补充要求 (documentation)：已通过单独提交补充了文档，PR 被批准并合并。

风险与影响

- 风险：兼容性风险：routed_experts_start_len 默认 0，与之前未使用该参数的行为一致，因此不会破坏现有 API。但需确保旧客户端未设置该字段时，服务端能够正确缺省为 0。

正确性风险：切片索引逻辑中的 start_len 需严格小于 seqlen-1，代码已有 min 防御；但若 start_len 过大导致返回空张量，调用方需要能够正确处理。测试中已覆盖越界场景。

性能风险：无，该变更仅在启用 return_routed_experts 的场景下生效，且默认路径（start_len=0）与之前完全一致。对于使用切片的请求，性能收益显著。

安全风险：无。

- 影响：对用户的影响：为 RL 训练框架（如使用 Kimi-K2、Qwen3-30B-A3B 的 MoE 模型）提供显著的性能优化，在最差情况下（32k 上下文）将路由数据收集延迟从 121ms 降至 1.9ms。用户需在客户端设置 routed_experts_start_len 为已累积的 prompt 长度，以获得增量路由数据。

对系统的影响：减少不必要的 GPU->CPU 数据搬运和 ZMQ 序列化负载，降低 decode 阶段的尾延迟。由于 DP 注意力机制，所有 DP rank 同步，本优化可整体提升系统吞吐。

对团队的影响：代码改动涉及请求生命周期多个模块，但均为简单的参数透传，理解和维护成本低。测试覆盖完整，降低了回归风险。

- 风险标记：向后兼容，参数边界验证，切片行为依赖调用方正确传参

关联脉络

- 暂无明显关联 PR