

PR #39038 完整报告

sgl-project/sglang

[Session] Work with PD and Fix empty continuations

合并时间: 2026-09-13 13:24

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39038>

执行摘要

- 一句话: 修复 session 在 PD 下丢失连接元数据与空续写失败
- 推荐动作: 值得精读, 重点是理解 session 与 PD 之间的元数据流: create_req 重建 Req 时为何必须透传 bootstrap 三元组与 worker ranks, 以及“输入校验放行 + 恢复历史后二次校验”的双层策略如何平衡 API 便利性与正确性。若你在维护 session 或多机推理入口, 可参考本 PR 的接线方式; 建议后续补一条空 continuation 的显式回归用例。

功能与动机

PR body 明确指出两个痛点: Session requests lose the connection fields needed by prefill and decode workers, 以及 Empty continuations fail before stored session history can supply the input。在 PD 部署下, session 请求由 session_controller 重建 Req 时未转发 bootstrap_host/port/room、worker ranks 与 disagg 模式, decode worker 拿到的 host 为 None, 构建连接地址时抛出 AttributeError: 'NoneType' object has no attribute 'startswith'; 而客户端依赖会话历史继续 (空输入) 的请求, 在 io_struct 的输入校验里就因 input_ids 为空被拒绝。

实现拆解

1. 输入校验放行 (python/sglang/srt/managers/io_struct.py): 在 _determine_batch_size 中, 当 input_ids 为空且 session_params 为 dict 并携带 id 时不再抛 ValueError, 而是把请求标记为单样本请求 (is_single=True, batch_size=1), 把“是否真的缺输入”的判断推迟到会话重建之后; 无 session id 的空输入仍按原逻辑报错, 保证普通请求不受影响。
2. 会话重建与二次校验 (python/sglang/srt/session/session_controller.py): Session.create_req 新增可选参数 disagg_mode (类型为 DisaggregationMode); 在 token 数组组装 (_share_token_arrays 或 _concat_token_arrays) 之后新增终审: 若当前未 abort 且组装后的 input_ids 仍为空, 则以 FINISH_ABORT 拒绝并提示 “A session request must contain input tokens after restoring history.”。构造新 Req 时补传 bootstrap_host/port/room、routed_dp_rank、disagg_prefill_dp_rank 与 disagg_mode, 保证 PD 下 prefill/decode worker 能拿到连接信息。
3. 调度器接线 (python/sglang/srt/managers/scheduler.py): 把默认 bootstrap 端口赋值从普通请求分支内提前到 handle_generate_request 入口统一执行, 使 session 请求同样获得默认端口; 在 session 分支调用 session.create_req 时传入

disagg_mode=self.disaggregation_mode。

4. 测试配套 (test/registered/unit/mem_cache/test_session_token_share_unit.py) : 仅给 _recv fixture 补上 bootstrap_host/port/room、routed_dp_rank、disagg_prefill_dp_rank = None 缺省字段, 保证单测构造的请求能通过新增的透传逻辑; 未新增针对空 continuation 或 PD 元数据透传的专门用例, PR 验证依赖 CPU component checks (main 分支 12 失败 → 修复后 20 全过) 与 46 个既有测试。

关键文件:

- python/sglang/srt/session/session_controller.py (模块 会话控制; 类别 source; 类型 entrypoint; 符号 Session.create_req) : 会话请求重建的核心入口: 新增 disagg_mode 参数、透传 bootstrap 连接字段与 worker ranks、并在恢复历史后对空输入做终审 abort, 是本 PR 的主修复点。
- python/sglang/srt/managers/io_struct.py (模块 输入校验; 类别 source; 类型 core-logic; 符号 TokenizedGenerateReqInput._determine_batch_size) : 输入校验主路径: 把空 input_ids 的硬校验改为“有 session id 时放行”, 是支持空 continuation 的前提, 与 session_controller 的终审形成双层校验。
- python/sglang/srt/managers/scheduler.py (模块 调度器; 类别 source; 类型 core-logic; 符号 Scheduler.handle_generate_request) : 调度路由入口: 默认 bootstrap 端口补齐从普通请求分支提前到函数入口, 使 session 请求也能拿到默认端口, 并向 create_req 传入 disaggregation 模式。
- test/registered/unit/mem_cache/test_session_token_share_unit.py (模块 会话测试; 类别 test; 类型 test-coverage; 符号 _recv) : 测试配套: 为 _recv fixture 补齐 transport 缺省字段, 避免新增透传逻辑后单测构造请求缺属性; 但未新增空 continuation 或 PD 元数据透传的专门用例。

关键符号: Session.create_req, TokenizedGenerateReqInput._determine_batch_size, Scheduler.handle_generate_request

关键源码片段

python/sglang/srt/session/session_controller.py

会话请求重建的核心入口: 新增 disagg_mode 参数、透传 bootstrap 连接字段与 worker ranks、并在恢复历史后对空输入做终审 abort, 是本 PR 的主修复点。

```
def create_req(
    self,
    req: TokenizedGenerateReqInput,
    tokenizer,
    vocab_size: int,
    eos_token_ids=None,
    disagg_mode: Optional[DisaggregationMode] = None,
):
    assert req.session_params is not None
    session_params = req.session_params

    # ... streaming 分支校验与 last_req 选择逻辑 ...
```

```

carry_fill = None
if last_req is not None:
    self._strip_bos_token(req, tokenizer)
    can_share_token_arrays = (
        self.streaming
        and self.committed_origin_len is not None
        and not session_params.drop_previous_output
        and not (session_params.offset and session_params.offset != 0)
    )
    if can_share_token_arrays:
        input_ids, input_ids_unpadded, carry_fill = self._share_token_arrays(
            last_req, req.input_ids
        )
    else:
        input_ids, input_ids_unpadded = self._concat_token_arrays(
            last_req, req, session_params
        )
else:
    # 会话内首个请求：没有历史可拼接，输入只能来自请求本身
    input_ids = req.input_ids
    input_ids_unpadded = req.input_ids

# io_struct 已放行空输入，这里基于真实历史做终审：
# 恢复历史后仍为空（如空首请求）必须拒绝，防止空序列进入执行队列
if not abort and len(input_ids) == 0:
    abort = True
    abort_message = (
        'A session request must contain input tokens after restoring history.'
    )

new_req = Req(
    rid=req.rid,
    origin_input_text=None,
    origin_input_ids=input_ids,
    origin_input_ids_unpadded=input_ids_unpadded,
    sampling_params=req.sampling_params,
    lora_id=req.lora_id,
    session=self,
    # PD 场景必需：decode / prefill worker 依赖这些字段构造连接地址，
    # 修复前 host 为 None 时会抛 AttributeError: 'NoneType' object has no attribute
    # 'startswith'
    bootstrap_host=req.bootstrap_host,
    bootstrap_port=req.bootstrap_port,
    bootstrap_room=req.bootstrap_room,
    disagg_mode=disagg_mode,
    routed_dp_rank=req.routed_dp_rank,
    disagg_prefill_dp_rank=req.disagg_prefill_dp_rank,
    # ... 其余字段与 session 登记逻辑保持不变 ...
)

```

python/sglang/srt/managers/io_struct.py

输入校验主路径：把空 input_ids 的硬校验改为“有 session id 时放行”，是支持空 continuation 的前提，与 session_controller 的终审形成双层校验。

```
elif self.input_ids is not None:
    if len(self.input_ids) == 0:
        # 空 continuation: 会话历史可能提供全部输入，这里仅放行；
        # 终审由 session_controller.create_req 基于恢复后的历史完成
        session_id = (
            self.session_params.get('id')
            if isinstance(self.session_params, dict)
            else None
        )
        if not session_id:
            raise ValueError('input_ids cannot be empty.')
        # 空输入按单样本处理，让请求进入会话重建流程
        self.is_single = True
        self.batch_size = 1
    elif isinstance(self.input_ids[0], int):
        self.is_single = True
        self.batch_size = 1
    else:
        self.is_single = False
        self.batch_size = len(self.input_ids)
    self.input_embeds = None
```

python/sglang/srt/managers/scheduler.py

调度路由入口：默认 bootstrap 端口补齐从普通请求分支提前到函数入口，使 session 请求也能拿到默认端口，并向 create_req 传入 disaggregation 模式。

```
def handle_generate_request(self, recv_req, *, mm_input_error=None):
    session_id = (
        recv_req.session_params.id if recv_req.session_params is not None else None
    )
    # 默认端口提前到路由入口统一补齐：
    # 修复前只在普通请求分支内生效，session 请求重建时端口为 None
    if recv_req.bootstrap_port is None:
        recv_req.bootstrap_port = get_disagg().disaggregation_bootstrap_port

    radix_native_session = (
        recv_req.session_id is not None and self.enable_session_radix_cache
    )

    if session_id is None or radix_native_session:
        # 普通请求或 radix-native session 请求：原有处理逻辑
        ...
    elif (
        session_id in self.session_controller
        and not self.session_controller.get(session_id).close_on_finish
```

```

):
    session = self.session_controller.get(session_id)
    req = session.create_req(
        recv_req,
        self.tokenizer,
        self.model_config.vocab_size,
        eos_token_ids=self.model_config.hf_eos_token_id,
        # 把 prefill / decode 分离模式透传给重建的 Req
        disagg_mode=self.disaggregation_mode,
    )
    ...

```

评论区精华

该 PR 无代码评审评论 (review_comments_count = 0)，讨论集中在 PR body 的验证记录与 CI 重跑。ByronHsu 在 body 中用两个场景说明修复效果：空 continuation 场景 “Before: [] -> input validation -> ERROR: input_ids cannot be empty. After: history + [] -> [1, 2, 3, 4] -> accepted”；session 元数据场景 “After: metadata -----> same host, room, ranks, and mode”，端口省略时落到默认值 8999。此外作者先后对 [test_disaggregation_decode_radix_cache.py](#) (8-gpu-h20) 与 session control / streaming session extra (1-gpu-h100) 发起 [/rerun-test](#) 均通过，并给出结论 “Failures are unrelated”，即 CI 失败被判定为与本次改动无关。

- 空 continuation 的放行与二次校验策略 (design): 采用“输入校验放行 + 会话重建后二次校验”的双层策略，保证空 continuation 可用且空首请求失败。
- PD 模式下 session 请求丢失 transport 元数据 (correctness): create_req 透传 bootstrap_host/port/room、routed_dp_rank、disagg_prefill_dp_rank 与 disagg_mode，调度器把默认端口补齐提前到路由入口，问题解决。
- CI 重跑与失败归属 (testing): 回归用例通过，剩余 CI 失败被判定为与本次改动无关。

风险与影响

- 风险:
 - 输入校验放宽的形态依赖: io_struct.py 里 session_id 仅在 session_params 为 dict 且含 id 时得到放行；若某些调用路径以对象形态传入 session_params（如既有单测用 SimpleNamespace），空输入仍会抛 ValueError，行为不对称，不同入口表现可能不一致。
 - 核心路由路径变更: scheduler.py 把默认 bootstrap_port 赋值提前到 handle_generate_request 入口，影响所有请求类型，需确认未启用 get_disagg() 时默认端口取值不会破坏非 PD 路径。
 - 调用点同步风险: Session.create_req 新增 disagg_mode 参数后，若未来新增调用点未传该参数，重建的 Req 会缺失 prefill/decode 模式信息，PD 场景下同样会退化。
 - 缺少新增回归测试: 测试仅补 fixture 缺省字段，没有为“空 continuation 接受”和“PD 元数据透传”新增专门单测或 e2e 用例，后续回归主要依赖 CPU component checks 覆盖。
- 影响:

- 用户 /API 层面：session 续写现在允许客户端只传 session id 而输入为空（依赖历史继续），PD 部署下 session 请求的连接元数据不再丢失，decode worker 可正常构建连接地址。
- 系统层面：非 session 请求的行为基本不变（仅端口填充位置提前）；session 请求新增 disagg 状态透传，radix-native session 走原有分支不受影响。
- 团队层面：修复了 CPU component checks 中 12 个失败用例，session 相关 46 个既有测试保持通过；改动范围小、单 commit，便于回溯。
- 风险标记：核心调度路径变更，输入校验放宽依赖调用方传参形态，缺少新增回归测试，disagg 元数据需调用点同步

关联脉络

- PR #39144 [Session + MM] Fix text positions in session continuations: 同为 session 续写正确性修复，改动同一功能线（session continuation 语义），可互证会话历史重建的边界条件。
- PR #39202 config: delete the redundant full stamp in initialize_model_parallel: 同期对 disaggregation / parallel 参数解析管线做钩子化重构，与本 PR 透传 disagg 元数据同属 PD 配置与状态传递链路的演进。