PR #31217 完整报告

sgl-project/sglang

[Disagg][StagingBuffer][1/2] Robustness and failure handling

合并时间：2026-07-24 17:22

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31217

执行摘要

- 一句话：修复 PD staging buffer 泄漏与注册竞争等健壮性问题
- 推荐动作：此 PR 值得精读，尤其 release_room 的设计体现了失败路径中资源清理的严谨思考，权衡了同步开销与数据安全。注册顺序调整解决了一个微妙的推拉竞争条件。建议关注第二部分 [2/2] RadixCache 的后续演进。

功能与动机

在构建 RadixCache 支持（[2/2]）过程中，发现了 staging buffer 路径的多个健壮性问题，每个问题均独立且可能导致永久性阻塞或资源泄漏。本 PR 将这些修复独立拆分以便审查。

实现拆解

1. 新增 release_room 释放逻辑：在 staging_handler.py 的 unregister_decode_req 中调用 release_room，同步 scatter stream 后释放未完成分配的 staging 槽位，避免 watermark 卡死。
2. 调整注册顺序：将 register_decode_req 调用从 decode.py 的 extend 移至 pop_preallocated，确保在 send_metadata 之前完成注册，消除 STAGING_REQ 丢弃窗口。
3. 失败显式传播：删除了 mooncake/conn.py 和 nixl/conn.py 中静默忽略 CHUNK_READY 发送失败的 try-except，改为直接抛出异常；同时 _do_staging_transfer 在 staging 无法容纳时返回 -1 而非回退，避免泄漏。
4. 启动时配置验证：在 prefill.py 的 __init__ 中添加对 chunked_prefill_size、pp_size、enable_prefill_context_parallel 的兼容性检查，不满足时直接报错。
5. 配套测试与文档更新：修改了两处测试文件以调整参数，删除文档中已废弃的环境变量说明。

关键文件：

- python/sglang/srt/disaggregation/common/staging_handler.py（模块 存根处理；类别 source；类型 entrypoint；符号 release_room, init_staging_buffers）：核心修改文件，新增 release_room 方法用于失败时释放未完成的 staging 分配，调整注册顺序和添加配置验证
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 Mooncake 后端；类别 source；类型 core-logic；符号 _send_chunk_ready, _do_staging_transfer, _init_staging_buffers）：Mooncake 传输后端，修改 _send_chunk_ready 移除静默异常捕

获，_do_staging_transfer 返回 -1 而非回退，并传入 chunked_prefill_size

- python/sglang/srt/disaggregation/nixl/conn.py（模块 NIXL 后端；类别 source；类型 dependency-wiring；符号 transfer_worker, _do_staging_transfer, _init_staging_buffers）：NIXL 传输后端，类似 mooncake 调整，导入 STAGING_WATERMARK_WAIT_S，修改 transfer_worker 和 _do_staging_transfer 以显式失败
- python/sglang/srt/disaggregation/prefill.py（模块 预填充；类别 source；类型 core-logic；符号 init）：启动时验证 staging 兼容配置，确保 chunked_prefill_size 页对齐、不支持 pp_size>1 和 context parallelism
- python/sglang/srt/disaggregation/decode.py（模块 解码调度；类别 source；类型 core-logic；符号 pop_preallocated, extend）：调整 register_decode_req 调用位置，在 pop_preallocated 中优先于 send_metadata 注册，避免 STAGING_REQ 竞争；清理 extend 中的重复注册

关键符号：release_room, init_staging_buffers, _send_chunk_ready, _do_staging_transfer, register_decode_req, unregister_decode_req, transfer_worker

关键源码片段

python/sglang/srt/disaggregation/prefill.py

启动时验证 staging 兼容配置，确保 chunked_prefill_size 页对齐、不支持 pp_size>1 和 context parallelism

```
# 在 PrefillRequest.__init__ 中，当启用 SGLANG_DISAGG_STAGING_BUFFER 时，
# 验证 staging 所需的服务端参数兼容性，不满足则立即报错。
if envs.SGLANG_DISAGG_STAGING_BUFFER.get():
    if self.is_mla_backend:
        raise RuntimeError(
            "SGLANG_DISAGG_STAGING_BUFFER 仅为非 MLA 模型设计"
            "（例如 GQA、MHA）。MLA 模型不应设置此标志。"
        )
    server_args = self.scheduler.server_args
    page_size = self.scheduler.token_to_kv_pool_allocator.page_size
    # chunked_prefill_size 是 staging 网格分块的基础：必须
    # 为正整数且是 page_size 的整数倍，否则无法划分固定网格。
    cps = server_args.chunked_prefill_size or 8192
    if cps <= 0 or cps % page_size != 0:
        raise RuntimeError(
            f"SGLANG_DISAGG_STAGING_BUFFER 要求 chunked_prefill_size "
            f"为正整数且是 page_size（{page_size}）的整数倍；"
            f"当前值：{server_args.chunked_prefill_size}。"
        )
    # Staging 写入方没有 pp（流水线并行）维度，不支持 pp_size > 1。
    if self.pp_size > 1:
        raise RuntimeError(
            "SGLANG_DISAGG_STAGING_BUFFER 不支持 pp_size > 1。"
        )
```

```
# 上下文并行会按 rank 重写 index_slice，破坏 chunk grid。
if server_args.enable_prefill_context_parallel:
    raise RuntimeError(
        "SGLANG_DISAGG_STAGING_BUFFER 不支持预填充上下文并行。"
    )
```

评论区精华

- release_room 同步必要性：YAMY1234 解释同步 scatter stream 是因失败路径已释放 KV-pool 页面，后续请求可能复用，必须同步防止数据竞争。结论：采用 per-event synchronize，仅阻塞调用线程。
- 是否 kill 传输线程：ShangmingCai 建议不应 kill 传输线程，因同一 prefill 可能服务其他 decode 实例。YAMY1234 接受并改为仅失败当前 room 并返回 -1。
- 运行时错误 vs 初始化检查：ShangmingCai 询问 nixl 中 handle is None 错误能否通过初始化配置避免。YAMY1234 指出 decode 侧 staging 大小在 bootstrap 时才可知，无法静态保证。
- chunked_prefill_size 验证：ShangmingCai 询问 --chunked-prefill-size -1 情况，YAMY1234 添加了启动时 guard 禁止该配置与 staging 同时使用。
  - release_room 中同步 scatter stream 的必要性 (correctness): 确认同步是必要的，采用 per-event synchronize 而非全设备同步，仅阻塞调用线程。
  - staging 失败是否应 kill 传输线程 (design): 改为 warning 并返回 -1，只失败当前 room。
  - 运行时错误 vs 初始化检查 (question): 保留运行时检查，因为动态协商无法静态验证。
  - chunked_prefill_size 配置验证 (correctness): 添加了启动时验证，chunked_prefill_size 必须为正整数且页对齐。

风险与影响

- 风险：
  - release_room 调用 stream.synchronize() 可能阻塞主线程，影响延迟，但仅在失败路径执行。
  - 注册顺序调整可能引入新的竞争条件，若 pop_preallocated 在其他场景提前调用可能导致未注册（但当前仅一处调用，风险可控）。
  - 启动时配置验证增加约束，可能影响已使用非兼容配置的用户部署（但限制合理）。
  - 删除文档中的环境变量可能使用户困惑，但变量已被取代。
- 影响：
  - 用户影响：PD 传输在失败 / 中止场景下更加稳健，不再有永久阻塞或泄漏，提升系统可靠性。
  - 系统影响：仅在失败路径增加少量同步开销，正常路径性能不变。重试机制避免了忙等待，减少 CPU 浪费。
  - 团队影响：为后续 RadixCache 支持奠定基础，代码结构更清晰，失败路径易于推理。
  - 风险标记：核心路径变更，并发安全，同步开销

关联脉络

- 暂无明显关联 PR