PR #36351 完整报告

sgl-project/sglang

fix(disagg): snapshot affected rooms before iterating outside the lock

合并时间：2026-08-26 03:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36351

执行摘要

- 一句话：锁外迭代前快照 affected rooms 修复竞态
- 推荐动作：值得精读，尤其是对 CommonKVManager 和节点故障处理机制感兴趣的工程师。该 PR 展示了如何通过简单的快照修复隐患，并详细分析了各后端的访问模式，对理解锁内外数据一致性有启发。

功能与动机

PR 描述指出，_handle_node_failure 在 connection_lock 下取出房间集合，但释放锁后才迭代。.get() 返回的是活集合而非副本，安全性依赖于所有变异方的访问模式，而 MoriKVManager._cleanup_room_tracking 违反了该假设，可能在同一对象上 discard 导致迭代期间修改。一旦触发，异常会逃逸心跳线程，导致节点故障检测永久静默失效，风险极高。尽管当前只有 Mooncake 和 Nixl 使用该线程，但 mori 接入仅需一行，且失效后果严重，因此值得加固。

实现拆解

1. 定位问题：在 python/sglang/srt/disaggregation/common/conn.py 的 _handle_node_failure 中，原代码 possible_affected_rooms = self.addr_to_rooms_tracker.get(failed_bootstrap_addr, []) 在锁内获取活集合，锁外迭代。
2. 修改方案：将获取改为 possible_affected_rooms = list(self.addr_to_rooms_tracker.get(failed_bootstrap_addr, [])），在锁内完成快照。list() 对内置集合是原子的，可防止迭代期间其他线程修改。
3. 行为保持不变：既访问相同房间，也照样 pop 跟踪条目，无需其他改动。
4. 测试与配置：PR 明确说明不添加单元测试（时间依赖的竞态难以确定性测试），仅提供复现实验数据；CI 阶段触发了 disaggregation 相关测试并全部通过。

关键文件：

- python/sglang/srt/disaggregation/common/conn.py（模块 连接管理；类别 source；类型 core-logic；符号 _handle_node_failure）：核心修改文件，修复锁外迭代活集合的竞态问题。

关键符号：_handle_node_failure

关键源码片段

python/sglang/srt/disaggregation/common/conn.py

核心修改文件，修复锁外迭代活集合的竞态问题。

```
# python/sglang/srt/disaggregation/common/conn.py ( 节选 )

def _handle_node_failure(self, failed_bootstrap_addr: str):
    """处理 prefill 节点故障。"""
    with self.connection_lock:
        keys_to_remove = [
            k for k in self.connection_pool if k.startswith(failed_bootstrap_addr)
        ]
        stale_endpoints = set()
        for k in keys_to_remove:
            for info in self.connection_pool[k]:
                ip = info.get("rank_ip")
                port = info.get("rank_port")
                if ip and port:
                    na = NetworkAddress(ip, int(port))
                    stale_endpoints.add(na.to_tcp())
        for k in keys_to_remove:
            del self.connection_pool[k]
        self.prefill_info_table.pop(failed_bootstrap_addr, None)

        # 注意：这里必须在锁内生成快照，不能直接引用活集合。
        # 因为后续迭代发生在锁外，而 `Mori` 后端可能缓存该集合的引用并
        # 在无锁情况下修改它，导致迭代时 `RuntimeError`。
        possible_affected_rooms = list(
            self.addr_to_rooms_tracker.get(failed_bootstrap_addr, [])
        )
        self.addr_to_rooms_tracker.pop(failed_bootstrap_addr, None)

    for endpoint in stale_endpoints:
        CommonKVReceiver.disconnect_endpoint(endpoint)

    affected_rooms = []
    for room in possible_affected_rooms:
        if (
            room in self.request_status
            and self.check_status(room) != KVPoll.Success
        ):
            self.record_failure(
                room,
                f"Lost connection with prefill instance (bootstrap_addr: {failed_bootstrap_addr})",
            )
            self.update_status(room, KVPoll.Failed)
            affected_rooms.append(room)

    logger.error(
        f"Lost connection with prefill instance (bootstrap_addr: {failed_bootstrap_addr}), "
```

```
        f"{len(affected_rooms)} requests affected"
    )
```

评论区精华

无 review 评论，但 PR 描述中作者主动分析了各后端的访问模式，指出 Mori 的 `_cleanup_room_tracking` 是唯一不安全的变异方，并强调该修复属于加固而非修复活 bug，同时验证了 `list()` 的原子性优于 Python 层级别理解。

- 暂无高价值评论线程

风险与影响

- 风险：风险低。改动仅涉及一行核心逻辑，行为不变，不会引入回归。但需注意：该修复依赖 list() 的原子性，若未来修改为其他快照方式（如理解 comprehension）可能重新引入竞态。此外，未覆盖 mori 接入场景的测试，若未来启用 mori 心跳线程，仍需验证。
- 影响：影响范围小，仅影响节点故障处理路径，属于控制面。对用户无感知，但能增强系统稳定性，避免故障检测静默失效。对团队而言，减少了跨后端的隐含依赖，便于未来接入 mori。
- 风险标记：核心路径变更，缺少测试覆盖，依赖 list 原子性

关联脉络

- PR #27010 [HiCache] Fix PP inconsistency with HiCache L3 (#22607): 同为 KV cache 管理相关修复，涉及调度与一致性，虽模块不同但可参考类似的锁与一致性处理。