PR #27796 完整报告

sgl-project/sglang

[PD] Fix ZMQ stale socket reconnection in PD disaggregation

合并时间：2026-06-11 19:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27796

执行摘要

- 一句话：修复 PD 分解中 ZMQ 陈旧套接字重连问题
- 推荐动作：建议精读，尤其是锁设计和 socket 生命周期管理。该 PR 展示了分布式系统中连接清理的最佳实践，但 mooncake 端的线程安全问题值得后续修复。

功能与动机

Issue 由 #27039 发现，本 PR 补充 *decode* 侧的修复。节点故障后，*decode* 侧未断开与故障 prefill 实例的 ZMQ 连接，导致后续重连时端口被占用或收到陈旧数据。

实现拆解

实现分四步：

1. 在 common/conn.py 的 _handle_node_failure 中，在删除 connection_pool 条目前，遍历连接信息（rank_ip+rank_port）构造 TCP endpoint 集合 stale_endpoints。
2. 完成原有清理逻辑后，遍历 stale_endpoints 调用新增的 CommonKVReceiver.disconnect_endpoint(endpoint) 类方法。
3. 新增 disconnect_endpoint 类方法：在全局锁保护下从 _socket_cache 和 _socket_locks 弹出 socket 和 lock，若存在则持有 lock 关闭 socket 并记录日志。同时为 _connect 方法添加 LINGER=0 和 RECONNECT_IVL_MAX 选项以改善重连行为。
4. 删除 mooncake/conn.py 中的 _handle_node_failure 方法（原为重复实现），使 MooncakeKVManager 继承基类 CommonKVManager 的处理逻辑。 测试配套：无新增测试文件，依赖现有 CI 测试。

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, disconnect_endpoint, _connect）：核心变更：增强 _handle_node_failure 收集 stale 端点并调用新增的 disconnect_endpoint 类方法清理 ZMQ socket；新增 disconnect_endpoint 方法并优化 _connect 选项。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 Mooncake 连接；类别 source；类型 core-logic；符号 _handle_node_failure）：删除重复的 _handle_node_failure 方法，使 MooncakeKVManager 复用基类 common/conn.py 中的统一逻辑，简化维护。

关键符号：_handle_node_failure, disconnect_endpoint, _connect

关键源码片段

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

核心变更：增强 _handle_node_failure 收集 stale 端点并调用新增的 disconnect_endpoint 类方法清理 ZMQ socket；新增 disconnect_endpoint 方法并优化 _connect 选项。

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

@classmethod
def disconnect_endpoint(cls, endpoint: str):
    """
    断开并清理指定端点的 ZMQ PUSH 套接字。
    在全局锁保护下安全关闭 socket，防止并发使用已关闭的 socket。
    """
    with cls._global_lock:
        # 从缓存中弹出 socket 及其关联锁
        sock = cls._socket_cache.pop(endpoint, None)
        lock = cls._socket_locks.pop(endpoint, None)
    if sock:
        # 若存在锁，则持有锁关闭，确保此时没有线程在发送
        if lock:
            with lock:
                sock.close()
        else:
            sock.close()
        logger.debug(f"Disconnected stale ZMQ PUSH socket (receiver): {endpoint}")
```

评论区精华

review 由 gemini-code-assist[bot] 提出多项线程安全建议：

- _connect 方法应返回 per-endpoint lock 以确保 socket 操作线程安全（已在 common/conn.py 的 _connect 中实现返回 sock, lock）。
- disconnect_endpoint 应在关闭 socket 前获取 per-endpoint lock（已在实现中满足）。
- 在 mooncake/conn.py 中的 send_aux_data_to_endpoint 和 sync_status_to_decode_endpoint 应获取 lock 后再发送（但最终 mooncake/conn.py 未改动这些方法，线程安全问题可能遗留）。
- 还指出 connection_pool 在 _setup_bootstrap_infos 中修改时未加锁，可能导致与 _handle_node_failure 的锁竞争（未解决）。 这些讨论表明 reviewer 关注并发正确性，但最终作者仅部分采纳。
- 线程安全：_connect 应返回 per-endpoint lock (design): 已采纳：common/conn.py 的 _connect 现已返回 sock 和 lock，调用方可以获取锁后发送。
- 线程安全：disconnect_endpoint 应持有 per-endpoint 锁 (design): 已采纳：disconnect_endpoint 在关闭 socket 前先获取 lock（如果存在），确保安全。

- 线程安全：mooncake/conn.py 的 send 方法应加锁 (design): 未解决：该 PR 未修改 mooncake/conn.py 的 send 方法，线程安全问题可能仍存在。
- Race condition: _setup_bootstrap_infos 修改 connection_pool 未加锁 (correctness): 未解决：此问题未被作者回复或修复，潜在风险。

风险与影响

- 风险：主要风险：
  1. 线程安全：mooncake/conn.py 中的 send 方法仍然直接调用 _connect 返回的 socket 而未加锁（因 mooncake 侧的 _connect 未返回 lock），可能导致并发 socket 竞争。基类 common/conn.py 的 _connect 返回 lock，但 mooncake 重写了 _connect 方法（在 mooncake/conn.py 中），它没有返回 lock，所以 send 时没有保护。这是一个潜在 bug。
  2. connection_pool 在 _setup_bootstrap_infos 中修改时未加锁，可能与 _handle_node_failure 或心跳线程冲突，造成迭代时修改错误。
  3. 测试覆盖不足：无针对性单元测试，依赖集成测试。 - 影响：影响 PD 分解模式下所有 decode 实例。修复可避免连接泄漏和端口冲突，但若存在线程安全问题，可能在极端并发时导致 crash 或数据错误。影响范围限于使用 disaggregation 特性的部署。 - 风险标记：线程安全：send 未加锁，并发竞争：connection_pool 写入无锁，缺少单元测试

关联脉络

- PR #27039 [EPD] fix: zmq PUSH socket reconnect-aware connection management with tcp keepalive: 本 PR 是 #27039 的补充，修复 decode 侧的重复连接问题。