PR #27039 完整报告

sgl-project/sglang

[EPD] fix: zmq PUSH socket reconnect-aware connection management with tcp keepalive

合并时间：2026-06-11 19:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27039

执行摘要

- 一句话：修复 EPD 中 ZMQ 连接失效后不断重试导致端口冲突
- 推荐动作：值得精读。该 PR 展示了 ZMQ 连接管理的几个最佳实践：通过 monitor socket 主动检测断连而非依赖超时；使用 RECONNECT_IVL=-1 接管重连控制；TCP keepalive 的三层参数配置。希望深入 ZMQ 编程或维护分离部署系统的工程师应关注。

功能与动机

在 EPD 分离部署中，当 Decode 节点重启或缩容时，Prefill 节点的 ZMQ 连接会持续发送 TCP SYN 包到历史 Decode 机器 IP，造成端口冲突；同样 Encode 节点也有此问题。旧版 @cache 装饰的 _connect 从不关闭或替换失效 socket，配合 ZMQ 的默认自动重连，导致无限 SYN 重试到不可达端点。

实现拆解

第一步：在 conn.py 的 CommonKVReceiver 中移除 functools.cache 并引入显式 socket 缓存 _socket_cache、monitor 缓存 _monitor_cache 和线程锁。 第二步：在 _connect 方法中，优先从缓存获取 socket，若存在则检查 monitor 是否有 EVENT_DISCONNECTED 事件，如有则关闭旧 socket 并清理缓存，随后创建新 socket。 第三步：创建新 socket 时设置 RECONNECT_IVL=-1 禁用自动重连、SNDTIMEO=30s 和 LINGER=0 避免无限阻塞，并启用 TCP keepalive（IDLE=30s, INTVL=5s, CNT=3）确保及时检测死连接。 第四步：在 encode_server.py 中将临时 socket 的 close() 改为 close(linger=5000)，限制 flush 等待时间。本次变更未包含测试文件，但 CI 承担了验证。

关键文件：

- python/sglang/srt/disaggregation/common/conn.py（模块 连接管理；类别 source；类型 dependency-wiring；符号 _connect, CommonKVReceiver）：核心变更文件，替换了 ZMQ PUSH socket 的连接管理方式，引入缓存、monitor、TCP keepalive 和禁用自动重连。
- python/sglang/srt/disaggregation/encode_server.py（模块 连接关闭；类别 source；类型 core-logic；符号 send_with_socket）：修改了临时 socket 的关闭方式，增加 linger 超时防止无限挂起

关键符号：_connect, send_with_socket

关键源码片段

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

核心变更文件，替换了 ZMQ PUSH socket 的连接管理方式，引入缓存、monitor、TCP keepalive 和禁用自动重连。

```
# _connect 方法：带缓存的 ZMQ PUSH socket 初始化与断连恢复
# 使用显式缓存 _socket_cache 替代 functools.cache，保证断开后可重建
# 通过 ZMQ monitor socket 监听 EVENT_DISCONNECTED 事件
with self._socket_lock:
    sock = self._socket_cache.get(endpoint)
    if sock is not None:
        monitor = self._monitor_cache.get(endpoint)
        disconnected = False
        if monitor is not None:
            try:
                # 非阻塞检查 monitor 是否有已就绪事件
                monitor.recv_multipart(zmq.NOBLOCK)
                disconnected = True
            except zmq.Again:
                pass # 无事件，连接仍正常
            except zmq.ZMQError:
                disconnected = True
        if not disconnected:
            return sock # 缓存有效，直接返回
        # 连接已断开，清理旧资源
        sock.close(linger=0)
        if monitor is not None:
            monitor.close()
        # 从缓存中移除旧条目（避免返回已关闭 socket）
        self._socket_cache.pop(endpoint, None)
        self._monitor_cache.pop(endpoint, None)

    # 创建新 socket
    sock = self._zmq_ctx.socket(zmq.PUSH)
    if is_ipv6:
        sock.setsockopt(zmq.IPV6, 1)
    # 禁用自动重连：由我们自己通过 monitor 控制重建
    sock.setsockopt(zmq.RECONNECT_IVL, -1)
    # 发送超时 30 秒，避免死等
    sock.setsockopt(zmq.SNDTIMEO, 30000)
    # LINGER=0：关闭时立即丢弃未发送数据，不阻塞
    sock.setsockopt(zmq.LINGER, 0)
    # 启用 TCP keepalive，三层参数细化检测
    sock.setsockopt(zmq.TCP_KEEPALIVE, 1)
    sock.setsockopt(zmq.TCP_KEEPALIVE_IDLE, 30)
    sock.setsockopt(zmq.TCP_KEEPALIVE_INTVL, 5)
    sock.setsockopt(zmq.TCP_KEEPALIVE_CNT, 3)
    sock.connect(endpoint)
    # 存入缓存
```

```
    self._socket_cache[endpoint] = sock
    # 创建 monitor socket 用于监听断开事件
    self._monitor_cache[endpoint] = sock.get_monitor_socket(zmq.EVENT_DISCONNECTED)
    return sock
```

python/sglang/srt/disaggregation/encode_server.py

修改了临时 socket 的关闭方式，增加 linger 超时防止无限挂起

```
# Encode 节点向 Prefill 节点发送 embedding 的临时 socket
# 每次 _send 新建一个 PUSH socket，发送后立即关闭
def send_with_socket():
    sock = self.sync_context.socket(zmq.PUSH)
    config_socket(sock, zmq.PUSH)
    try:
        sock.connect(endpoint)
        if buffer is not None:
            sock.send_multipart([serialized_data, buffer], copy=False)
        else:
            sock.send_multipart([serialized_data], copy=False)
    finally:
        # close(linger=5000)：最多等待 5 秒刷新数据
        # 若对端已消失，不会无限阻塞，5 秒后直接丢弃
        sock.close(linger=5000)
```

评论区精华

1. Critical bug（gemini-code-assist）：在连接恢复中若 socket 重建失败，旧 socket 未被从 _socket_cache 移除，可能返回已关闭 socket。作者后续调整了清理顺序（先 pop 再创建）予以解决。
2. Medium 建议（gemini-code-assist）：建议复用同一个 zmq.Context 实例，避免多个 context 浪费资源。
3. _disconnect 未调用（liusy58）：询问 _disconnect 是否从未被调用，作者回应“done”，但最终代码中可能已移除该方法。
4. encode_server 是否需要 RECONNECT_IVL（liusy58）：作者解释临时 socket 生命周期短，不需要禁用自动重连，且与缓存 socket 设计解耦。最终 ShangmingCai 批准合并。

- 连接恢复中 socket 缓存清理 bug (correctness): 作者最终调整清理顺序，在创建新 socket 前先 pop 旧缓存，问题已解决。
- 建议复用 ZMQ Context 实例 (performance): 作者将局部 context 替换为 self._zmq_ctx 并在初始化时创建，供后续复用。
- _disconnect 方法未被使用 (style): 作者回复 'done'，最终代码中可能移除了该方法的定义或调整了调用路径。
- encode_server 是否需要 RECONNECT_IVL (design): 作者解释临时 socket 生命周期短（每次新建），不需要禁用自动重连；且与缓存 socket 设计解耦。

风险与影响

- 风险：
  1. 边缘情况处理：若在 socket 重建过程中 zmq.Context.socket 或 get_monitor_socket 抛出异常，缓存虽已清理但未创建新 socket，下次调用可能重试，但不排除遗留线程安全问题（锁保护已到位但异常时可能状态不一致）。
  2. TCP keepalive 平台差异：TCP_KEEPALIVE_IDLE/INTVL/CNT 的默认值与 Linux 内核相关，其他平台行为需验证。
  3. 无新增测试：未配套添加单元测试或集成测试，回归风险依赖现有 CI。
  4. 内存泄漏隐患：如果 monitor socket 未被正确关闭（但本 PR 已确保关闭），可能造成 fd 泄漏。 - 影响：影响范围：所有使用 EPD 分离部署的 Prefill→Decode 和 Encode→Prefill 通信路径。 用户影响：修复了节点弹性伸缩时的端口冲突，提升部署可靠性，避免了无效 SYN 包对训练 / 推理机器 NCCL 端口的污染。 系统影响：减少网络垃圾包，提高连接恢复确定性。 团队影响：连接管理逻辑复杂度提升，后续维护者需理解 monitor 和 keepalive 机制。
- 风险标记：核心路径变更，缺少测试覆盖，TCP keepalive 平台依赖

关联脉络

- PR #27696 [RL] Handle Mooncake buffers across memory release: 同属分离部署模块，改进连接管理的稳定性。