PR #24671 完整报告

sgl-project/sglang

fix(nixl): close file descriptors after each FILE transfer

合并时间：2026-05-13 15:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24671

执行摘要

- 一句话：修复 NIXL FILE 传输文件描述符泄漏
- 推荐动作：该 PR 是一个典型的资源泄漏修复案例，设计上采用最小变更 + 回归测试的策略值得参考。建议关注作者后续的重构 commit，以了解如何系统性解决 NixlFileManager 的资源管理问题。

功能与动机

PR 描述指出：NixlFileManager.files_to_nixl_tuples() 在每次 batch_set_v1/batch_get_v1 调用中为每个 key 通过 os.open() 打开一个文件描述符，但从未在成功路径上关闭。实测观察到 fd 数量持续增长（例：一次 batch_set 后 open_fds 从 167 增至 1351），最终可能导致系统文件描述符耗尽。

实现拆解

实现拆解：

1. 保护 _execute_transfer 中的 FILE 分支：在 hicache_nixl.py 的 _execute_transfer 方法中，将原有的、未受保护的 FILE 分支及其后的传输逻辑整体移入 try/finally 块。在 try 入口初始化 file_fds = []，在调用 files_to_nixl_tuples 成功后立即将所有打开的描述符存入 file_fds。
2. 无条件关闭描述符：在 finally 块中遍历 file_fds，逐一调用 file_manager.close_file(fd)。无论函数正常返回、早 return False、或抛出异常，均确保执行关闭。
3. 修复测试基础框架：测试文件 test_hicache_nixl_storage.py 首先针对上游 HiCacheStorageConfig 新增的必需参数（pp_rank、pp_size、attn_cp_rank、attn_cp_size、enable_storage_metrics、extra_config）补齐调用，避免 test setup 失败。同时移除已废弃的 plugin 参数。
4. 新增文件描述符泄漏检测：在测试类中添加静态方法 _open_fds() 用于查询当前进程的 fd 数量。在 test_mixed_operations 各操作前后插入断言 self.assertEqual(self._open_fds(), fds, "fd leak after xxx")，验证操作不会导致 fd 增长。
5. 整理测试用例：将错误的 test_register_files_with_tuples 重命名为正确的 test_register_files，修正调用签名（传入文件路径而非 NIXL 元组）。

关键文件：

- python/sglang/srt/mem_cache/storage/nixl/hicache_nixl.py（模块 存储层；类别 source；类型 dependency-wiring）：核心修复文件，在 _execute_transfer 方法中添加 try/finally 确保文件描述符关闭。
- python/sglang/srt/mem_cache/storage/nixl/test_hicache_nixl_storage.py（模块 存储层测试；类别 test；类型 test-coverage；符号 _open_fds, test_register_files_with_tuples, test_register_files）：回归测试文件，新增 _open_fds 方法和 fd 泄漏断言，并修复测试套件以适配上游配置变更。

关键符号：_execute_transfer, _open_fds, test_register_files_with_tuples, test_register_files

关键源码片段

python/sglang/srt/mem_cache/storage/nixl/hicache_nixl.py

核心修复文件，在 _execute_transfer 方法中添加 try/finally 确保文件描述符关闭。

```
# python/sglang/srt/mem_cache/storage/nixl/hicache_nixl.py

    def _execute_transfer(
        self,
        buffers: Optional[List[torch.Tensor | tuple]],
        keys: List[str],
        direction: str,
    ) -> bool:
        if len(buffers) != len(keys):
            logger.error("Mismatch between number of tensors/buffers and files/objects")
            return False

        # file_fds 列表用于收集所有打开的描述符，确保在 finally 中被关闭
        file_fds = []
        try:
            if self.backend_selector.mem_type == "FILE":
                # files_to_nixl_tuples 为每个 key 打开一个 fd，返回的 tuple 第 3 个元素是 fd
                tuples = self.file_manager.files_to_nixl_tuples(keys)
                file_fds = [t[2] for t in tuples]
                if not tuples or not self.registration._register_memory(tuples, "FILE"):
                    logger.error("Failed to prepare files for transfer")
                    return False
            else: # mem_type == "OBJ"
                tuples = [(0, 0, key, "") for key in keys]
                if not tuples or not self.registration._register_memory(tuples, "OBJ"):
                    logger.error("Failed to register objects")
                    return False

            # ... 后续传输准备和执行逻辑（不变） ...

        finally:
            # 无条件关闭本次调用中打开的所有文件描述符
```

```
            for fd in file_fds:
                self.file_manager.close_file(fd)
```

python/sglang/srt/mem_cache/storage/nixl/test_hicache_nixl_storage.py

回归测试文件，新增 _open_fds 方法和 fd 泄漏断言，并修复测试套件以适配上游配置变更。

```
# python/sglang/srt/mem_cache/storage/nixl/test_hicache_nixl_storage.py

    @staticmethod
    def _open_fds() -> int:
        # 通过读取 /proc/self/fd 获取当前进程打开的文件描述符数量，用于泄漏检测
        return len(os.listdir("/proc/self/fd"))

    def test_mixed_operations(self):
        # ...setup...
        self.assertTrue(self.hicache.set(key1, value1))
        # 首次 set 后记录基准 fd 数（吸收任何一次性内部打开）
        fds = self._open_fds()
        retrieved1 = self.hicache.get(key1, dst1)
        self.verify_tensors_equal(value1, retrieved1)
        # 验证 get 后 fd 数不变
        self.assertEqual(self._open_fds(), fds, "fd leak after get")

        # Batch set/get
        self.assertTrue(self.hicache.batch_set([key2], [value2]))
        self.assertEqual(self._open_fds(), fds, "fd leak after batch_set")
        retrieved2 = self.hicache.batch_get([key2], [dst2])
        self.verify_tensors_equal(value2, retrieved2[0])
        self.assertEqual(self._open_fds(), fds, "fd leak after batch_get")
```

评论区精华

Review 中 gemini-code-assist[bot] 提出了三点改进建议：

- PEP8 导入位置：import traceback 不应出现在函数体内，应置于文件顶部。
- try/finally 范围过宽：OBJ 内存类型不涉及文件描述符，也被包裹在内，增加了缩进层级，建议使用上下文管理器收窄范围。
- register_files 可能仍有泄漏：register_files 方法同样调用了 files_to_nixl_tuples 但没有关闭 fd，构成潜在泄漏点。

作者 lluki 回复表示该修复有意保持最小改动以确保稳定性，后续将提交专门的重构 commit 解决更大范围的问题。维护者 xiezhq-hermann 最终批准合并。

- PEP8 导入位置 (style): 作者未回应，但此问题不影响功能，后续重构可一并修正。PR 已合并，此点未解决。
- try/finally 范围过宽 (design): 作者回复称当前保持最小改动确保稳定，后续会提交重构 commit 解决。审查者同意后合并。

- register_files 潜在泄漏 (correctness): 此点未被作者当面回应，但当前 PR 已合并。register_files 泄漏将在后续重构中统一处理。

风险与影响

- 风险：风险较低：
  - 修改集中在 _execute_transfer 一个函数内，通过 try/finally 保证文件描述符关闭，不会改变原有的成功 / 失败路径逻辑。
  - 新增的测试断言可能偶发失败（如果系统 fd 数因外部原因变化），但 _open_fds 读取 /proc/self/fd 是快照，非原子，中间有其他线程可能影响，但测试环境通常可控。
  - register_files 方法仍有相同泄漏问题未被本次修复覆盖，但该函数使用场景不同（仅注册不传输），暂未被触发生产问题，已列入后续重构计划。
  - 影响：影响范围局限于使用 NIXL POSIX FILE 后端的 HiCache 用户。修复后，长时间、高吞吐的 KV 缓存传输不再因文件描述符耗尽而崩溃，系统稳定性显著提升。测试增强为后续重构提供了回归保障。对 OBJ 和 DRAM 后端无影响。
  - 风险标记：资源泄漏修复，测试覆盖增强

关联脉络

- 暂无明显关联 PR