PR #7157 完整报告

verl-project/verl

[worker] fix: close temporary TransferQueue event loop

合并时间：2026-07-27 15:50

原文链接：http://prhub.com.cn/verl-project/verl/pull/7157

执行摘要

- 一句话：修复 TransferQueue 临时事件循环 FD 泄漏
- 推荐动作：值得精读，特别是事件生命周期管理和资源清理模式。PR 非常精准地修复了一个隐蔽的文件描述符泄漏问题，变更量小但影响大。提出的全局事件循环重构建议值得跟进，但当前修复是正确的紧急修补。

功能与动机

关联 Issue #7156 报告了长期运行分布式训练中因 TransferQueue 临时事件循环未关闭导致文件描述符泄漏的问题。Issue 复现显示 32 次 _run_async_in_temp_loop 调用泄漏 96 个 FD，生产环境大约 70 个训练步骤后触发 OSError: [Errno 24] Too many open files 和后续的 Gloo 连接重置。需要在不破坏现有临时循环架构的前提下关闭事件循环。

实现拆解

1. 关闭临时事件循环：在 verl/utils/transferqueue_utils.py 的 _run_async_in_temp_loop 函数中，于 finally 块内、线程 join() 之后添加 tmp_event_loop.close() 调用，确保每次循环都释放 selector 文件描述符。
2. 简化停止循环方式：将原本通过创建 stop_loop 协程并提交到事件循环的方式，改为直接使用 tmp_event_loop.call_soon_threadsafe(tmp_event_loop.stop)，避免额外创建协程并减少间接性。同时移除了不再需要的 stop_loop 异步函数定义。
3. 添加回归测试：新增 tests/utils/test_transferqueue_utils_on_cpu.py，测试 test_temp_event_loop_releases_file_descriptors：在禁用循环 GC 的条件下重复调用 _run_async_in_temp_loop 32 次，通过 /proc/self/fd 计数验证文件描述符无增长；仅在有 /proc 的 Linux 系统上运行。

关键文件：

- verl/utils/transferqueue_utils.py（模块 工具库；类别 source；类型 core-logic；符号 _run_async_in_temp_loop）：核心修改文件，修复了 _run_async_in_temp_loop 中的事件循环泄漏。添加了 call_soon_threadsafe 停止循环和 close() 调用。
- tests/utils/test_transferqueue_utils_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _noop, _open_fd_count, test_temp_event_loop_releases_file_descriptors）：新增的 Linux CPU 回归测试，验证 32 次 _run_async_in_temp_loop 调用后文件描述符数量不增长。

关键符号：_run_async_in_temp_loop, _open_fd_count, test_temp_event_loop_releases_file_descriptors

关键源码片段

verl/utils/transferqueue_utils.py

核心修改文件，修复了 _run_async_in_temp_loop 中的事件循环泄漏。添加了 call_soon_threadsafe 停止循环和 close() 调用。

```
def _run_async_in_temp_loop(async_func: Callable[..., Any], *args, **kwargs) -> Any:
    # Use a temporary event loop in a new thread because event
    # loop may already exist in server mode
    tmp_event_loop = asyncio.new_event_loop()
    thread = threading.Thread(
        target=tmp_event_loop.run_forever,
        name="batchmeta tensordict converter",
        daemon=True,
    )

    def run_coroutine(coroutine):
        if not thread.is_alive():
            thread.start()
        future = asyncio.run_coroutine_threadsafe(coroutine, tmp_event_loop)
        return future.result()

    try:
        return run_coroutine(async_func(*args, **kwargs))
    finally:
        # 顺序：先让事件循环停止 -> 等待线程退出 -> 关闭循环，释放文件描述符
        if thread.is_alive():
            # 使用 call_soon_threadsafe 替代自定义协程，更直接高效
            tmp_event_loop.call_soon_threadsafe(tmp_event_loop.stop)
            thread.join()
        # 关闭循环，释放 selector 等资源 —— 这是之前遗漏的关键步骤
        tmp_event_loop.close()
```

tests/utils/test_transferqueue_utils_on_cpu.py

新增的 Linux CPU 回归测试，验证 32 次 _run_async_in_temp_loop 调用后文件描述符数量不增长。

```
async def _noop():
    return None

def _open_fd_count() -> int:
    # 仅在 Linux 上有 /proc/self/fd 时运行，否则跳过测试
    if not os.path.isdir("/proc/self/fd"):
        pytest.skip("requires Linux /proc file descriptor accounting")
    return len(os.listdir("/proc/self/fd"))
```

```
def test_temp_event_loop_releases_file_descriptors():
    gc.collect()
    gc_was_enabled = gc.isenabled()
    gc.disable() # 禁用循环 GC，确保 FD 泄漏仅由事件循环未关闭导致，而非 GC 延迟
    try:
        before = _open_fd_count()
        for _ in range(32):
            _run_async_in_temp_loop(_noop)
        leaked = _open_fd_count() - before
    finally:
        if gc_was_enabled:
            gc.enable()
        gc.collect()
    # 断言 32 次调用后无 FD 增长
    assert leaked == 0
```

评论区精华

设计讨论：Reviewer wuxibin89 建议创建一个全局事件循环和线程，而不是每次调用都创建新循环。对此，0oshowero0 回复同意，并承诺后续会重构设计。但当前 PR 作为最小修复保留了临时循环架构，仅追加关闭操作以解决泄漏问题。

- 临时循环 vs 全局事件循环 (design): 0oshowero0 同意后续将重构为全局事件循环设计，但当前 PR 作为最小修复保留现有架构。

风险与影响

- 风险：低风险：变更仅影响 _run_async_in_temp_loop 的清理路径，且行为是确定性的：先 call_soon_threadsafe 停止循环，再 join 线程，最后 close 循环。与原代码相比，唯一新增的是 close() 调用，而 close() 在循环已停止后调用是安全的。回归测试验证了 32 次调用后无 FD 泄漏。潜在风险：如果 tmp_event_loop.close() 在循环仍在运行时被调用（不会，因为之前有 stop + join），或事件循环已关闭导致异常（通常不会），但代码已确保正确顺序。
- 影响：影响范围：直接影响所有使用 TransferQueue 进行 BatchMeta/TensorDict 转换的 worker 进程（包括 rollout worker 和 trainer）。在长期运行的多节点训练中，此修复将消除因文件描述符耗尽导致的进程崩溃和分布式通信故障（如 Gloo 连接重置）。API、配置或命令行行为无变化。现有用户可通过更新代码获得提升的稳定性，无需修改配置。
- 风险标记：事件循环资源管理，缺少全局循环重构

关联脉络

- PR #7158 [rollout] fix: TypeError merging numpy routed_experts on partial rollout resume: 同系列 rollout/transferqueue 相关 bugfix，涉及 workers/rollout 模块，同作者 le-czs。
- PR #7139 [sglang] fix: use _base guard in _compact_for_bucket to prevent NCCL buffer race: 另一个涉及 rollout 中资源竞争的 bugfix，同属稳定性和资源管理改进。

- PR #7144 [ckpt, fsdp] feat: sharded delta block placements + backend-owned HF export (engine core + FSDP): 同期影响 worker 和 rollout 引擎的较大变更，可能引入更多 TransferQueue 调用。