PR #24601 完整报告

sgl-project/sglang

[PD] Centralize per-room cleanup in common backend

合并时间：2026-05-07 18:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24601

执行摘要

- 一句话：集中化 PD 每房间清理逻辑到公共基类
- 推荐动作：本 PR 是高质量的代码整洁重构，消除了重复代码和隐式副作用，降低了维护成本和竞态风险。建议团队成员精读 common/conn.py 中新增的 clear() 实现，理解如何安全地在基类中统一清理各后端状态。同时建议采纳 gemini-code-assist 的 review 建议，修复 Nixl 后端的潜在内存泄漏问题，并补充 addr_to_rooms_tracker 的清理。

功能与动机

Per-room state cleanup 在多个后端和多处调用点分散实现，容易遗漏字段（如 addr_to_rooms_tracker 只在 mori receiver 中清理），update_status(Failed) 携带的 req_to_decode_prefix_len pop 副作用导致调用脆弱，Nixl 中直接 del request_status/transfer_infos 使后续 clear() 调用会引发 KeyError。本 PR 旨在统一清理入口，消除重复、遗漏和竞态风险。

实现拆解

1. 在公共基类中添加统一 clear() 方法：在 python/sglang/srt/disaggregation/common/conn.py 中，为 CommonKVSender 和 CommonKVReceiver 分别新增 clear() 方法，使用 pop(..., None) 安全清除所有 per-room 字典条目。
2. 移除 update_status 的隐式清理副作用：从 CommonKVManager.update_status 中删除对 req_to_decode_prefix_len.pop(bootstrap_room, None) 的调用，该职责完全转移到统一的 clear() 中。
3. 删除后端子类中的重复 clear() 实现：移除 MooncakeKVSender.clear()、MooncakeKVReceiver.clear()、MoriKVSender.clear() 中的重复清理逻辑（它们现在是基类实现的子集）。
4. 简化 mori receiver clear 为 super 调用 + 后端特有清理：将 MoriKVReceiver.clear() 简化为 super().clear() 加上 mori 特有的 room_to_bootstrap_addr 清理。
5. 修复 Nixl 后端的竞态风险：将 Nixl 中 add_transfer_request 和 send 的 del 语句替换为 pop(..., None)，并移除 send 中直接删除 request_status 的行，使后续统一 clear() 调用不会 KeyError。
6. 测试与CI：无新增测试文件，依赖现有disintegration测试验证，通过rerun触发并通过。

关键文件：

- python/sglang/srt/disaggregation/common/conn.py（模块 PD 分解；类别 source；类型 core-logic；符号 clear）：核心变更文件，在基类中新增统一 clear() 方法，并移除 update_status() 的隐式清理副作用，是所有后端清理的基础。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 PD 分解；类别 source；类型 core-logic；符号 clear）：删除 MooncakeKVSender.clear() 和 MooncakeKVReceiver.clear() 的重复实现，直接继承基类方法。
- python/sglang/srt/disaggregation/mori/conn.py（模块 PD 分解；类别 source；类型 core-logic；符号 clear）：删除 MoriKVSender.clear()（与基类完全相同），简化 MoriKVReceiver.clear() 为 super().clear() + 后端特有清理。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 PD 分解；类别 source；类型 core-logic）：将 Nixl 中 del 替换为 pop，移除 send() 中对 request_status 的强行删除，避免与统一 clear() 冲突。

关键符号：clear, update_status, failure_exception, send, add_transfer_request

关键源码片段

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

核心变更文件，在基类中新增统一 clear() 方法，并移除 update_status() 的隐式清理副作用，是所有后端清理的基础。

```
# python/sglang/srt/disaggregation/common/conn.py
# 在 CommonKVSender 和 CommonKVReceiver 中新增统一 clear() 方法

class CommonKVSender(BaseKVSender):
    # ... 其他方法

    def clear(self) -> None:
        """清除所有 per-room 状态，使用 pop(..., None) 避免 KeyError。"""
        self.kv_mgr.request_status.pop(self.bootstrap_room, None)
        # req_to_decode_prefix_len 和 transfer_infos 并非所有后端都有，用 hasattr 保护
        if hasattr(self.kv_mgr, "req_to_decode_prefix_len"):
            self.kv_mgr.req_to_decode_prefix_len.pop(self.bootstrap_room, None)
        if hasattr(self.kv_mgr, "transfer_infos"):
            self.kv_mgr.transfer_infos.pop(self.bootstrap_room, None)

    def abort(self):
        self.kv_mgr.record_failure(
            self.bootstrap_room,
            "Aborted by AbortReq.",
        )
        self.kv_mgr.update_status(self.bootstrap_room, KVPoll.Failed)
        self.conclude_state = KVPoll.Failed


class CommonKVReceiver(BaseKVReceiver):
    # ... 其他方法
```

```
def clear(self) -> None:
    """清除所有 per-room 状态，适用于 receiver 端的字段。"""
    self.kv_mgr.request_status.pop(self.bootstrap_room, None)
    self.kv_mgr.required_prefill_response_num_table.pop(self.bootstrap_room, None)
    self.kv_mgr.prefill_response_tracker.pop(self.bootstrap_room, None)
```

评论区精华

gemini-code-assist[bot] 在审核中提出两点改进建议：

- Nixl 后端内存泄漏风险：在 python/sglang/srt/disaggregation/nixl/conn.py 的 send() 中，移除 del self.kv_mgr.request_status[self.bootstrap_room] 后未调用 self.clear()，可能导致 request_status 字典内存泄漏。建议在该处添加 self.clear() 调用。
- CommonKVReceiver.clear() 缺少 addr_to_rooms_tracker 清理：建议在 CommonKVReceiver.clear() 中补充对 addr_to_rooms_tracker 的清理，并添加 bootstrap_room is None 的安全检查。 截至分析时，这两条建议尚未被采纳或关闭。
- Nixl 后端内存泄漏风险 (performance): 未解决。分析时尚未采纳建议。
- CommonKVReceiver.clear() 缺少 addr_to_rooms_tracker 清理 (correctness): 未解决。分析时尚未采纳建议。

风险与影响

- 风险：
  1. 回归风险：移除各后端子类的重复 clear() 实现后，如果基类 clear() 未能覆盖所有应清理的字典（如 addr_to_rooms_tracker），可能导致内存泄漏。gemini-code-assist 的 review 已指出此风险。
  2. Nixl 后端内存泄漏：gemini-code-assist 指出 NixlKVSender.send() 中移除 del request_status 后未调用 self.clear()，可能导致 request_status 字典持续增长。需确认是否已在其他路径（如 failure_exception 或 abort）中调用了 clear()。
  3. 并发安全：pop(..., None) 是线程安全的，但 hasattr 检查之后可能被其他线程修改？当前 kv_mgr 对象在 clear() 调用期间应该不会被并发修改，风险较低。
  4. Ascend 后端继承 Mooncake 的自动修复：MooncakeKVSender 的 clear() 被删除后，AscendKVSender（继承自 Mooncake）将自动使用基类 CommonKVSender.clear()，但需确认 Ascend 是否有额外需要清理的字段。
- 影响：
  - 用户影响：无直接用户可见变化，但持续运行的 PD 分解服务中内存泄漏风险降低，稳定性提升。
  - 系统影响：清理逻辑集中化后，新增后端或新增 per-room 状态字段时不易遗漏，维护成本降低。
  - 团队影响：消除了 update_status(Failed) 的隐式副作用，使状态更新语义更纯净，未来调用 update_status 时不再意外触发清理。
  - 影响范围：涉及 4 个后端（common、mooncake、mori、nixl），其中 Nixl 和 Mori 的清理语义发生了变化，需要验证。

- 风险标记：Nixl 后端内存泄漏风险，缺少 addr_to_rooms_tracker 清理，缺少 bootstrap_room None 检查

关联脉络

- PR #24550 [R3] Avoid implicit CUDA sync in routed experts DP slicing: 同为 PD 分解相关修复，涉及后端清理和状态管理。
- PR #24005 [AMD] Enable dual-stream MoE on ROCm: 涉及 mooncake 后端（moriep.py）的变更，可能与本 PR 的 mooncake 清理逻辑调整有关联。