PR #35049 完整报告

sgl-project/sglang

[PD] Deferred decode-side KV release for aborts mid-transfer

合并时间：2026-08-18 23:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35049

执行摘要

- 一句话：PD 传输中途 abort 的 KV 延迟释放，默认关闭
- 推荐动作：值得精读。这是典型的并发正确性修复 PR，设计论证质量高：单写者论证（一个 room 只由唯一 transfer worker 写与 ack）、dummy-proof 分母、先提交存活集合再逐个释放的事务式 resolve、以及“迟到 ack 不污染复用房间”的防御，都是值得借鉴的模式。若团队正在使用 PD 解耦并遇到偶发错答，应重点评估开启该特性的条件与 30 秒超时假设。

功能与动机

PR body 明确指出：abort 一个 decode 请求（长 ITL abort、超时或 /abort_request）时，若其 prefill→decode 的 KV 传输仍在途，系统会立即释放该请求的 KV 页面和 req-pool 槽位，但不会取消传输；下一个请求可能拿到这些页面，被上一次的迟到写入覆盖，造成“下一个请求静默持有第一个请求的 KV”且全程无任何报错。这是 @ShangmingCai 在 #27372 中预见的竞态（原文："we could introduce deferred release of decode-side KVCache to fix this"），也是 open issue #32564 的核心症状；PR 还指出这种竞态使 #34876 的 LIFO 立即复用“最大化 free→reuse 窗口”而不安全。

实现拆解

实现分为 5 个步骤：

1. decode 侧延迟持有（decode.py）：DecodeTransferQueue 新增 enable_deferred_kv_release 与 deferred_kv_release_timeout 两个配置字段，并维护 _deferred_releases 列表。在 pop_transferred 的 KVPoll.Failed 分支中，若开启特性且 kv_receiver.abort_notified（decode 发起的 abort），调用 _defer_release 将请求移入 hold 列表，并跳过该轮 metadata 释放；resolve_deferred_releases 在 process_decode_queue 与 PP 路径的 process_decode_transfer_queue 中无条件提前调用，按 ack 或超时释放。核心不变式是 required_acks = len(bootstrap_infos)（所有被通知的 prefill rank），而不是 required_prefill_response_num，避免 dummy rank 造成提前释放。
2. prefill 侧 ABORT_ACK 语义改造（mooncake/conn.py）：bootstrap 线程收到 ABORT 时，对 active room 先 update_status(Failed)（阻止 add_transfer_request 继续入队新 chunk）再注册 _deferred_ack_targets，不再由 bootstrap 线程直接 ack；_send_abort_ack 携带 _prefill_unique_rank()（按 attn_tp/pp/attn_cp 计算的稳定 sender id）作为 ack 的 rank 标识。房间所属的唯一 transfer worker 在 skip 与

completion 两个站点调用 _maybe_ack_drained_abort，当 _staging_outstanding[room] == 0 时发送 ACK，靠 pop 保证只触发一次。

3.排空语义保障（mooncake/conn.py）：drain-ack依赖send_kvcache返回时无在途写入。默认同步路径本就满足；隐患在 custom-mem-pool executor 的失败分支——原先遇到第一个失败 future 就 cancel() 其余并返回，而 cancel() 对已运行的 future 是 no-op，sibling 写入可能晚于 chunk 结论。三处 KV 写 executor loop 统一改为走 _await_transfer_futures，开启时先 drain 全部 running futures 再返回，关闭时保持原 early-return 行为。

4. 调度与泄漏 invariant 配套（scheduler.py + scheduler_pp_mixin.py）：abort_request 在发送 ABORT 的同时 register_deferred_abort_room 武装 ack tracker，捕获下一个 forward step 前到达的 ACK；Scheduler.on_idle 的池泄漏检查在存在 pending holds 时跳过（否则 idle 时会误报 pool leak 并杀掉 decode 调度器），scheduler_pp_mixin.py 保证 resolve_deferred_releases 与 release_rids 解耦，内存压力下超时仍能触发。

5. 配置与测试（environ.py + 两个测试文件）：新增 SGLANG_DISAGGREGATION_DEFERRED_DECODE_KV_RELEASE（默认 False）与 SGLANG_DISAGGREGATION_DEFERRED_DECODE_KV_RELEASE_TIMEOUT（默认 30.0）；新增 test_deferred_decode_kv_release.py 覆盖 ack 聚合、重复 rank 不重复计数、单 rank 快路径、迟到 ack 不污染复用房间、失败释放隔离等场景，并修补 test_decode_queue_cleanup.py 中绕过 __init__ 的测试夹具。

关键文件：

- python/sglang/srt/disaggregation/mooncake/conn.py（模块 传输层；类别 source；类型 core-logic；符号 _await_transfer_futures, _prefill_unique_rank, _send_abort_ack, _maybe_ack_drained_abort）：prefill 侧核心改造：ABORT 处理顺序改为先 Failed 再注册 ack 目标，ABORT_ACK 从“听到 abort”变为“传输已排空”，新增 _await_transfer_futures 保证 executor 失败分支也排空 sibling 写入，是整条 deferral 链路正确性的关键。
- python/sglang/srt/disaggregation/decode.py（模块 解码调度；类别 source；类型 core-logic；符号 _defer_release, _do_release, has_pending_deferred_releases, resolve_deferred_releases）：decode 侧核心改造：Failed 轮询时不再立即释放，而是 _defer_release 加入 hold 列表；resolve_deferred_releases 每 step 提前调用，按全部 rank ack 或超时释放，是延迟释放机制的主控逻辑。
- python/sglang/srt/disaggregation/common/conn.py（模块 连接管理；类别 source；类型 core-logic；符号 register_deferred_abort_room, note_abort_ack, is_abort_release_safe, clear_deferred_abort_state）：提供 per-room 的 ack 聚合原语：register_deferred_abort_room、note_abort_ack、is_abort_release_safe、clear_deferred_abort_state，并处理 bootstrap_room 复用与迟到 ack 污染问题。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：abort_request 在发送 ABORT 的同时武装 ack tracker，捕获下一个 forward step 前到达的 ACK；on_idle 在 holds 挂起时跳过池泄漏检查，修复实测崩溃。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：新增两个 env：SGLANG_DISAGGREGATION_DEFERRED_DECODE_KV_RELEASE（默认

False）与超时设置，是整个特性默认关闭、零影响的前提。

- python/sglang/srt/managers/scheduler_pp_mixin.py（模块 调度器；类别 source；类型 core-logic）：PP 路径的 process_decode_transfer_queue 无条件调用 resolve_deferred_releases，保证 ack/ 超时驱动的释放在内存压力下不被 release_rids 门限跳过。
- test/registered/unit/disaggregation/test_deferred_decode_kv_release.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_manager, TestAbortAckAggregation, test_release_safe_only_after_all_required_ranks_ack, test_duplicate_rank_ack_does_not_over_count）：新增 10 个单元测试，覆盖 ack 聚合计数、重复 rank、单 rank 快路径、迟到 ack 不污染复用房间、释放失败隔离等本 PR 最关键的防御性行为。
- test/registered/unit/disaggregation/test_decode_queue_cleanup.py（模块 单元测试；类别 test；类型 test-coverage）：修补绕过 __init__的测试夹具，显式设置 enable_deferred_kv_release=False，保证既有清理测试在特性关闭路径继续通过。

关键符号：_await_transfer_futures, _prefill_unique_rank, _send_abort_ack, _maybe_ack_drained_abort, _defer_release, _do_release, has_pending_deferred_releases, resolve_deferred_releases, register_deferred_abort_room, note_abort_ack, is_abort_release_safe, clear_deferred_abort_state

关键源码片段

python/sglang/srt/disaggregation/decode.py

decode 侧核心改造：Failed 轮询时不再立即释放，而是 _defer_release 加入 hold 列表；resolve_deferred_releases 每 step 提前调用，按全部 rank ack 或超时释放，是延迟释放机制的主控逻辑。

```
def _defer_release(self, decode_req: DecodeRequest) -> None:
    # 计算超时截止时间：默认 30 秒，与 mooncake 单次 transfer 上限对齐
    deadline = time.monotonic() + self.deferred_kv_release_timeout
    # 分母必须是“本次 abort 实际通知到的 prefill rank 数”（bootstrap_infos 长度）
    # 而不是 required_prefill_response_num：每个被通知的 rank 只会 ack 一次，到达
    # 该计数即证明没有任何 prefill 仍在写这些页面
    required_acks = len(decode_req.kv_receiver.bootstrap_infos)
    self._deferred_releases.append(
        (decode_req, deadline, decode_req.metadata_buffer_index, required_acks)
    )


def resolve_deferred_releases(self) -> None:
    """每个 scheduler step 早期调用：所有 prefill rank 确认排空或超时后释放。"""
    if not self._deferred_releases:
        return
    now = time.monotonic()
    still_held, to_release = [], []
```

```
    for decode_req, deadline, idx, required_acks in self._deferred_releases:
        room = decode_req.req.bootstrap_room
        kv_mgr = decode_req.kv_receiver.kv_mgr
        drained = kv_mgr.is_abort_release_safe(room, required_acks)
        if not drained and now < deadline:
            still_held.append((decode_req, deadline, idx, required_acks))
        else:
            to_release.append((decode_req, idx, room, drained))
    # 先提交"存活集合"再逐个释放：一旦 _do_release 抛异常，也不会留下
    # 已释放但仍在本列表中的条目，避免重复释放 metadata idx 或后续重试
    self._deferred_releases = still_held
    for decode_req, idx, room, drained in to_release:
        if not drained:
            logger.warning(
                f"Deferred KV release for room {room} timed out after "
                f"{self.deferred_kv_release_timeout}s without a full drain "
                f"ack from prefill; releasing anyway",
            )
        try:
            self._do_release(decode_req, idx)
        except Exception:
            # 隔离单个失败，保证其余条目仍被释放；条目已从列表移除，不会重试
            logger.exception(f"Deferred KV release failed for room {room}")
```

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

提供 per-room 的 ack 聚合原语：register_deferred_abort_room、note_abort_ack、is_abort_release_safe、clear_deferred_abort_state，并处理 bootstrap_room 复用与迟到 ack 污染问题。

```
def register_deferred_abort_room(self, bootstrap_room: int) -> None:
    # 用全新 set 武装房间；同时清掉上一个复用该 bootstrap_room 的请求
    # 留下的陈旧 ack，防止"迟到 ack"污染下一个请求
    self._deferred_abort_ack_tracker[bootstrap_room] = set()


def note_abort_ack(self, bootstrap_room: int, prefill_rank: int) -> None:
    # 只有房间正处于 hold 状态时才记账；先取引用再 add，避免与 clear 竞争
    acks = self._deferred_abort_ack_tracker.get(bootstrap_room)
    if acks is not None:
        acks.add(prefill_rank)


def is_abort_release_safe(self, bootstrap_room: int, required_acks: int) -> bool:
    # required_acks 来自 decode 侧实际通知到的 prefill rank 数；集合大小到达
    # 该值即证明没有 prefill 仍在写这些页面
    return (
        len(self._deferred_abort_ack_tracker.get(bootstrap_room, ()))
        >= required_acks
    )
```

评论区精华

该 PR 没有外部 human review（review_comments_count 为 0），但 PR body 用大量篇幅记录了作者自身的对抗性多智能体自审和 2×H200 实测验证，核心讨论点包括：

- prefill 处理顺序修正：自审发现若先注册 ack target 再标记 Failed，worker 可能在 room 尚未 Failed 时完成 drain 并 ACK，导致新入队 chunk 继续写入已释放页面，因此改为"先 Failed 后注册"。
- bootstrap_room 复用污染：bootstrap_room 非全局唯一，迟到 ACK 可能满足后续复用同一房间的请求并提前释放其页面；改为每次 abort 事件重新武装 / 复位 tracker，未注册房间的 ACK 直接丢弃。
- executor 失败分支的 sibling future：原路径在首个失败 future 上 return 并仅 cancel() 其余 future，正在运行的写入会继续落盘；新增 _await_transfer_futures 在特性开启时先 drain 再返回。
- idle 泄漏 invariant 误杀：实测发现持有期间 idle 检查会误报 pool leak 并崩溃 decode 调度器，通过 on_idle 跳过检查解决，并在 hold 全部消解后确认无泄漏。
- prefill abort 处理顺序：先 Failed 还是先注册 ack target (correctness): 改为先 update_status(Failed) 阻止新 chunk 入队，再注册 _deferred_ack_targets；ACK 由房间唯一 worker 在排空后发出。
- bootstrap_room 复用导致的迟到 ack 污染 (correctness): ack tracker 改为每次 abort 事件在 abort_request 发送 ABORT 时武装 / 复位，未注册房间的 ack 直接丢弃。
- executor 失败分支的 sibling future 未排空 (correctness): 三处 KV 写 executor loop 统一走 _await_transfer_futures，特性开启时先 drain 全部 running futures 再返回。
- idle 时池泄漏 invariant 误杀 decode 调度器 (correctness): on_idle 仅在 exists pending holds 时跳过池 /req-pool 泄漏检查，resolve 完成后恢复检查。
- fast-ACK 路径有效性未得到充分验证 (testing): PR body 明确标注为待办：需要更大模型 / 长上下文验证 fast-ACK 实际触发；安全性由超时兜底保证，当前不阻塞合入。

风险与影响

- 风险：主要技术风险集中在以下四点：

1. 多节点 /TP>1/dummy-rank fan-out 未验证：len(bootstrap_infos) 分母在真实异步多队列并发下未测，多节点 3 节点 H200 验证由 #32564 完成，本 PR 只在 2×H200 1P1D-1TP 场景验证，dummy rank 与多队列并发仍有理论不确定性。
2. 超时兜底依赖 30 秒假设：释放以 _staging_outstanding == 0 或超时为条件，超时值对齐 #27372 提到的 mooncake transfer 约 30 秒上限；若真实部署的传输时长超过该值，超时释放仍可能落在在途写入之前，虽然概率低但属于设计边界。
3. hold 资源无上限：abort storm 期间被 hold 的页面、req-slot、metadata idx 不设数量上限，最坏情况下 KV 池容量下降，等待请求退避或返回 500；这是修复损坏与可用性之间的显式权衡。
4. 开启后失败路径行为差异：关闭时 _await_transfer_futures 保留原 first-error early-return，开启时改为一并 drain 全部 futures，失败 chunk 的返回延迟会变长，可能

影响对崩溃 / 超时语义的既有假设。此外，feature 默认关闭意味着只有显式开启的部署才获得保护，未开启部署仍暴露于 #32564 竞态。 - 影响：影响范围限于 PD 解耦推理（mooncake 后端）的 abort 路径。对用户：开启后 abort 请求的 KV 不再被立即复用，从根源上消除“下一个请求静默持有上一个请求 KV”的错答；代价是 abort 后资源最多滞留 30 秒，极端场景下表现为背压 /500 而非错误的生成内容，属于正确的权衡方向。对系统：prefill 与 decode 两侧必须同时开启该 env 才生效，属部署级协调变更；调度器在每个 step 无条件调用 resolve_deferred_releases，开启后增加每轮轻量遍历。对团队：该 PR 是 #32564 完整 lease/barrier 方案之外的轻量替代，两者存在设计重叠，最终取舍是维护者决策。 - 风险标记：核心路径并发变更，默认关闭需显式开启，多节点 /TP>1 场景未验证，hold 资源无上限，超时兜底依赖 30 秒假设，CI 未全量通过，fast-ACK 路径缺乏实证

关联脉络

- PR #27372 PD abort 相关：防止 prefill 在 abort 后启动新工作：本 PR 的 ABORT_ACK TODO 源头；#27372 阻止 prefill 在 abort 后开始新工作，本 PR 处理已入队的在途传输，双 PR 共同覆盖 abort 竞态。
- PR #32564 [PD] Don't release KV pages while Mooncake transfers are in flight: 本 PR 要修复的核心 bug 来源（open issue），也是更重的 lease/barrier 方案；本 PR 是其轻量替代且存在设计重叠，最终取舍是维护者决策。
- PR #34876 O(1) tail-pop free-slot 变更：本 PR 明确指出是 #34876 安全落地的前置条件：LIFO 立即复用会最大化 free→reuse 窗口，必须先有延迟释放才能安全复用空闲槽位。