PR #35281 完整报告

sgl-project/sglang

[PD] Align defensive protocol behavior across Mooncake, NIXL, and Mori

合并时间：2026-08-31 10:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35281

执行摘要

- 一句话：对齐三 PD 后端防御性协议行为，修复竞态与 KV 布局缺陷
- 推荐动作：值得精读。这是 RFC #33861 驱动的 staged refactor 的标杆 PR：1）范围控制——主动回退与 #35049/#35360 冲突的畸形消息隔离，避免扩大 scope；2）review 深度——rishabhsinha17 逐分支对照 Common planner 核验，并抓出 NIXL guard 差一行和 _staging_ctx 未关闭竞态两个真实问题；3）合并策略——遇到上游已落地等价实现时保留上游代码、丢弃本地重复分支，而不是强行 rebase 冲突。建议后续跟进两个遗留项：将测试合入主仓库、关闭 _staging_ctx 的 WATERMARK 竞态。

功能与动机

RFC #33861 指出三个 PD 传输后端（Mooncake、NIXL、Mori）各自完整实现了 bootstrap + transfer 控制协议，但行为已漂移，"each copy is missing something the others have"：例如 Mori 的 Decode manager 从不启动 heartbeat checker，死掉的 Prefill peer 永远不会被检测到；Mori 在验证 MORI_GUARD 之后才处理 Common 协议发出的无 guard ABORT 帧，导致 ABORT 被当作畸形消息丢弃。跟踪 Issue #34510 把 " 低风险行为对齐 " 列为 Step 1，要求以多个小 PR 填补 Common 语义缺口，本 PR 是其中一环（另含 #34692、#36160、#36612、#37077）。PR body 明确说明范围：只补缺失行为或修复局部竞态 / 布局问题，不改变协议提取的后续阶段工作。

实现拆解

本 PR 按五个区域推进（其中第 3 项最终回退），核心原则是 " 只改后端局部、不动 Common 状态机 "。

1. Mori Decode heartbeat 与 ABORT 投递：python/sglang/srt/disaggregation/mori/conn.py 的 MoriKVManager.__init__ DECODE 分支新增 self._start_heartbeat_checker_thread()，复用 Common 已有的 heartbeat 检查线程，使死 Prefill 能触发房间失败与缓存驱逐。ABORT 识别方面，由于上游 #29133 已落地带 transfer_lock 的等价 handler，合并时保留上游实现，本分支的 _handle_abort_notification 被丢弃。
2. NIXL/Mori 丢弃 teardown 后的残留 chunk：nixl/conn.py 的 transfer_worker 在出队后先检查 room not in self.request_status，再以 transfer_infos.get(room) 替换原来的 assert room in self.transfer_infos，命中清除 _staging_outstanding 后 continue，避免 stale chunk 跌入断言异常路径；Mori 因 chunk 执行在 Sender 的 _run_chunk()，采用等

价 early return。

3. Mori KV 布局对齐 Common planner：_get_mha_mem_desc_slices 增加 same-PP 快速路径（双方 descriptor 等长时直接本地切分，不再套全局层偏移），并为 Decode 侧带 draft KV 的场景（[K_main..., V_main..., draft_K..., draft_V...]）按层数倍率计算 V 偏移；_get_mla_mem_desc_slices 增加 same-PP 快速路径；send_kvcache 分支条件由 is_mla_backend 扩展为 is_mla_backend or is_hybrid_mla_backend。
4. Mooncake 控制线程初始化顺序：mooncake/conn.py 将 session_failures、failed_sessions、session_lock 三个字段的赋值移到 start_prefill_thread() 之前，关闭 Decode 注册消息与线程启动的竞态窗口。
5. 畸形控制消息隔离（已回退）：原计划为 Mooncake/NIXL 控制循环加 per-message 异常边界，但 #35049/#35360 已引入更复杂的 deferred-release 与 ABORT/ABORT_ACK 状态转移，混入会扩大 scope，最终从 diff 中移除，留待协议统一阶段处理。
6. 测试配套：CPU red/green 回归测试（Mori ABORT、cleared-room stale work、畸形消息隔离、Mori speculative MHA 选择、Mooncake 初始化顺序）放在作者分支 jambow0320/sglang@rfc-pd-test 的 test/registered/unit/disaggregation/rfc-test/ 下验证，未合入主仓库。

关键文件：

- python/sglang/srt/disaggregation/mori/conn.py（模块 传输层；类别 source；类型 core-logic；符号 MoriKVManager.init, MoriKVManager._get_mha_mem_desc_slices, MoriKVManager._get_mla_mem_desc_slices, MoriKVManager.send_kvcache）：更改最集中的文件：补上 Mori Decode 缺失的 heartbeat 检测，将 MHA/MLA descriptor 切片逻辑对齐 Common planner（same-PP 快速路径 + draft KV V 偏移修正），并把 send_kvcache 扩展到 hybrid MLA 场景；也是 review 中 ABORT/ABORT_ACK 讨论的焦点。
- python/sglang/srt/disaggregation/nixl/conn.py（模块 传输层；类别 source；类型 core-logic；符号 NixlKVManager.transfer_worker）：transfer_worker 增加 cleared-room 与 metadata 双重守卫，消除 stale chunk 在断言路径上异常退出 worker 的风险；rishabhsinha17 指出并验证了该窗口的修复全过程。
- python/sglang/srt/disaggregation/mooncake/conn.py（模块 传输层；类别 source；类型 core-logic；符号 MooncakeKVManager.init）：将 session_failures/failed_sessions/session_lock 初始化前移到 start_prefill_thread() 之前，修复控制线程启动竞态；但 review 指出的 _staging_ctx 竞态在最终合并版本中仍未关闭。

关键符号：MoriKVManager.init, MoriKVManager._get_mha_mem_desc_slices, MoriKVManager._get_mla_mem_desc_slices, MoriKVManager.send_kvcache, NixlKVManager.transfer_worker, MooncakeKVManager.init

关键源码片段

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

更改最集中的文件：补上 Mori Decode 缺失的 heartbeat 检测，将 MHA/MLA descriptor 切片逻辑对齐 Common planner（same-PP 快速路径 + draft KV V 偏移修正），并把

send_kvcache 扩展到 hybrid MLA 场景；也是 review 中 ABORT/ABORT_ACK 讨论的焦点。

```
def _get_mha_mem_desc_slices(
    self, dst_mem_descs: List[MemoryDesc]
) -> tuple[
    List[MemoryDesc], List[MemoryDesc], List[MemoryDesc], List[MemoryDesc], int
]:
    src_descs = self.kv_mem_descs
    if not src_descs:
        raise RuntimeError("KV memory descriptors are empty on prefill side")

    num_local_layers = len(src_descs) // 2
    src_k_descs = src_descs[:num_local_layers]
    src_v_descs = src_descs[num_local_layers:]

    # 同 PP 场景下双方暴露的是 PP-local 布局，descriptor 索引本就对齐；
    # 若再套用 Prefill 侧的全局层偏移，会错误索引到本地列表之外。
    if len(src_descs) == len(dst_mem_descs):
        dst_k_descs = dst_mem_descs[:num_local_layers]
        dst_v_descs = dst_mem_descs[num_local_layers:]
        return (
            src_k_descs,
            src_v_descs,
            dst_k_descs,
            dst_v_descs,
            num_local_layers,
        )

    start_layer = self.kv_args.prefill_start_layer
    end_layer = start_layer + num_local_layers
    dst_total_layers = len(dst_mem_descs) // 2
    if len(dst_mem_descs) < 2 or end_layer > dst_total_layers:
        raise ValueError(
            "Destination KV descriptors do not match prefill pp configuration"
        )
    dst_k_descs = dst_mem_descs[start_layer:end_layer]

    # Decode 侧带 draft 模型 KV 而 Prefill 侧只有 target 模型 KV 时，
    # 目标布局为 [K_main..., V_main..., draft_K..., draft_V...]。
    # V 区间起点不再是 dst_total_layers，而需按层数倍率换算；
    # 否则会把 draft 区域的 descriptor 错当成主模型 V 来写。
    if (
        num_local_layers < dst_total_layers
        and dst_total_layers % num_local_layers != 0
    ):
        multiplier_ratio = dst_total_layers // num_local_layers
        dst_v_offset = num_local_layers * multiplier_ratio
    else:
        dst_v_offset = dst_total_layers
```

```
    dst_v_descs = dst_mem_descs[
        dst_v_offset + start_layer : dst_v_offset + end_layer
    ]
    return src_k_descs, src_v_descs, dst_k_descs, dst_v_descs, num_local_layers
```

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

transfer_worker 增加 cleared-room 与 metadata 双重守卫，消除 stale chunk 在断言路径上异常退出 worker 的风险；rishabhsinha17 指出并验证了该窗口的修复全过程。

```
def transfer_worker(self, queue: FastQueue, staging_buffer=None, worker_index=0):
    # 每个 worker 私有的 staging 策略：首个 chunk 时懒创建，
    # 以便看到 ModelRunner 在引擎初始化后设置的 kv_buffer_tensors；
    # 且不缓存在 self 上（多 worker 会竞争 ring）。
    staging_strategy = None

    while True:
        kv_chunk: TransferKVChunk = queue.get()
        room = kv_chunk.room
        handles: List[Any] = []
        try:
            # room 已被 Sender 清理时，队列中残留的 chunk 直接丢弃。
            # 此前缺失的 room 会绕过 Failed 检查，跌入 transfer_infos
            # 断言路径而异常退出 worker 循环。
            if room not in self.request_status:
                logger.debug(
                    "Skipping chunk for room %s because it has been cleared",
                    room,
                )
                self._staging_outstanding.pop(room, None)
                continue

            # 出队时计数（状态检查之前）：outstanding == 0 表示既无出队
            # 也无在飞传输——这是 abort ack 依赖的判定条件；defer 重入队
            # 时该标志依然保留，因此不会重复计数。
            if not kv_chunk.staging_counted:
                self._staging_outstanding[room] += 1
                kv_chunk.staging_counted = True

            if self.check_status(room) == KVPoll.Failed:
                self._staging_outstanding.pop(room, None)
                if self.enable_deferred_decode_kv_release:
                    # 跳过即未写入任何数据，对已 abort 的 room 直接 ack。
                    self._maybe_ack_drained_abort(room)
                continue

            # 用 dict.get 替换原先的 assert：conclude 成功后 transfer_infos
            # 会被 pop，而 request_status 仍保留 Success 直到 Sender clear()，
            # 延迟重入队的 stale chunk 若走 assert 会直接异常。
            room_transfer_infos = self.transfer_infos.get(room)
```

```
            if room_transfer_infos is None:
                logger.debug(
                    "Skipping chunk for room %s because its transfer metadata "
                    "has been cleared",
                    room,
                )
                self._staging_outstanding.pop(room, None)
                continue
            # ... 后续按 room_transfer_infos 处理请求列表
```

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

将 session_failures/failed_sessions/session_lock 初始化前移到 start_prefill_thread() 之前，修复控制线程启动竞态；但 review 指出的 _staging_ctx 竞态在最终合并版本中仍未关闭。

```
if self.disaggregation_mode == DisaggregationMode.PREFILL:
    # 这三个 session 状态字段必须在控制线程启动前就位；
    # 否则 Decode 注册消息恰好在线程启动窗口内到达时，
    # 会命中未定义的 session_failures / failed_sessions / session_lock。
    self.session_failures = defaultdict(int)
    self.failed_sessions = set()
    self.session_lock = threading.Lock()
    self.start_prefill_thread()
    # 注意：_staging_ctx 仍在 start_prefill_thread() 之后赋值，
    # review 指出 WATERMARK 消息在该窗口到达仍会命中缺失属性（未修复）。
```

评论区精华

Review 有两条主线：ShanglingCai 的协议语义追问，以及 rishabhsinha17 对照 Common planner 的逐点核验。

- Mori 是否需要 abort ack（ShangmingCai）：作者承认最终需要，但 MoriKVManager 没有 _staging_outstanding，共享 drain-ack helper 无法直接复用；完整 ack 需要 prefill 侧注册 ack 目标、Mori _start_decode_thread 识别 ABORT_ACK，本 PR 只做第一步（识别 ABORT），ack 路径留给后续 PR。
- heartbeat 是否依赖 #36160（ShangmingCai）：作者澄清 #36160 只改 Mori PREFILL 分支，本改动在 DECODE 分支，仅接线已有 Common heartbeat 检查器，两者无依赖。
- NIXL guard 差一行（rishabhsinha17）：成功 conclude 块会 pop transfer_infos 而 request_status 保留 Success 直到 clear()，延迟 staging 重入队的 stale chunk 仍能穿过新守卫。后续 revision 以 transfer_infos.get(room) 替换 assert 后关闭了该窗口。
- Mooncake _staging_ctx 竞态未关闭（rishabhsinha17）：start_prefill_thread() 启动后 _staging_ctx 才在 line 242 赋值，WATERMARK 消息在窗口内到达会命中缺失属性；最终合并版本中仍未修复。
- Mori 是否需要 ABORT ACK (design): 本 PR 不实现 ack；ABORT_ACK 与 quiescence 语义推迟到协议统一阶段定义。
- Mori heartbeat 是否依赖 #36160 (question): 无依赖，两者互不冲突。

- NIXL stale-room 守卫差一行：conclude 与 Success 状态窗口 (correctness): 已解决，审查者在 8388c351 后的复读中确认 Area 2 resolved。
- Mooncake _staging_ctx 初始化竞态仍未关闭 (correctness): 未解决：合并版本中 _staging_ctx 仍在控制线程启动后赋值，留有理论竞态窗口。
- 畸形控制消息隔离的回退决策 (design): 回退；畸形帧隔离与协议统一阶段合并处理，PR body 已知悉与最终 diff 的差异。

风险与影响

- 风险：
  1. Mooncake 初始化竞态未完全关闭：_staging_ctx 仍在 start_prefill_thread() 之后赋值（mooncake/conn.py 约 line 242 之前），若启用 staging 且 WATERMARK 控制消息在窗口内到达，handle_watermark_msg(self._staging_ctx, ...) 会访问未定义属性。review 已明确指出但最终未修复。
  2. 测试未合入主仓库：全部 CPU red/green 回归测试位于作者个人分支，主仓库无对应用例，上述守卫与布局修复后续回归风险较高。
  3. ABORT/ABORT_ACK 语义悬空：Mori Prefill 现在能识别 ABORT 并标记房间 Failed，但 decode 侧不发送 ack、不定义 quiescence 语义；未来引入 ACK 时需处理双解析窗口。
  4. 畸形控制消息隔离回退：Mooncake/NIXL 长生命周期控制线程仍可能被畸形帧终止，属于已记录但未修复的遗留问题，推迟到协议统一阶段。
  5. same-PP 快速路径前提依赖：Mori MHA/MLA 快速路径依赖 " 双方 descriptor 等长 " 这一假设，与 Common planner 的行为逐分支一致，风险较低，但需在后续提取 Common helper 时固化为契约。 - 影响：影响范围集中在 PD 分离（Prefill/Decode disaggregation）的三个 RDMA 后端传输路径：Mori（heartbeat、ABORT、speculative MHA/MLA 布局、hybrid MLA）、NIXL（transfer_worker 稳定性）、Mooncake（控制线程启动时序）。对用户而言，修复了死 Prefill peer 长期不被发现、abort 不足时请求悬挂、room 清理后 worker 异常退出、Mori speculative/hybrid 场景 KV 写错位置等实际缺陷。对团队而言，本 PR 是三后端行为收敛的示范：刻意缩小 scope、回退冲突改动、在合并冲突时保留上游 #29133 实现，为后续 Step 2 的 Per-backend Transport 提取和 Step 4 的 Common 协议集成铺平了道路。 - 风险标记：已指出的未关闭竞态：_staging_ctx，测试未合入主仓库，ABORT_ACK 语义待后续定义，畸形控制帧隔离被回退，PD 核心路径跨三后端改动

关联脉络

- PR #34692 NIXL Prefill bootstrap timeout: 前序 Step 1 PR，本 PR 在其基础上继续推进；PR body 明确列出为此前已完成项。
- PR #29133 Mori ABORT handler (upstream): 合并 main 时发现上游已落地等价实现，最终保留上游并由本 PR 丢弃本地重复代码。
- PR #36160 Mori PREFILL 侧 Step 1 PR: 与 #35281 同属 Step 1；review 中确认与本 PR 的 DECODE heartbeat 改动无依赖。

- PR #34977 Mori from_zmq 帧布局 truth-table 测试：review 中确认本 PR 的 ABORT 帧读取不与其帧布局测试冲突，可独立落地。