PR #37094 完整报告

sgl-project/sglang

[mem_cache] Move `req_pool_idx` into `ReqKvInfo`

合并时间：2026-08-31 05:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37094

执行摘要

- 一句话：req_pool_idx 收进 ReqKvInfo，引入 is_held，纯重构
- 推荐动作：值得精读。它是“无行为变更的大范围结构重构”的范本，可学习三点：一是如何用静态字段删除实现 fail-fast 防遗漏；二是如何用两步 commit 分离“搬迁”与“语义替换”；三是如何用统一假对象（_FakeOwner）消解测试对内部字段的依赖。重点关注 ReqKvInfo 中 req_pool_idx、is_held、is_released、mark_released 四者之间的关系，以及 streaming_session.py 中 save/restore 因字段收敛而简化的写法。

功能与动机

PR body 明确说明：Move req_pool_idx from Req / SessionSlot into ReqKvInfo, so the record that tracks a request's device KV also carries the req_to_token row that registers it, and let the record answer its own presence: ReqKvInfo.is_held (req_pool_idx is not None) replaces the identical Req.is_holding_kv / SessionSlot.is_holding_kv wrappers, pairing with the existing is_released。其目的是让“跟踪设备 KV 的记录”同时携带“注册它的 req_to_token 行”，使存在性判断不再跨 Req、SessionSlot、ReqKvInfo 三个对象拼凑字段；同时 Follows up on #37078，是 KV ownership 系列重构的延续。

实现拆解

1. 核心数据结构收敛（schedule_batch.py）：ReqKvInfo 新增 req_pool_idx: Optional[int] = None，注释明确其为 req_to_token 的注册行；新增只读属性 is_held，与既有 is_released 配对。Req.__init__ 删除 self.req_pool_idx，删除 Req.is_holding_kv 属性；reset_for_retract 的断言、offload_kv_cache / load_kv_cache 的取行逻辑全部切换。静态字段删除使任何残留读点立即 AttributeError，形成 fail-fast 防线。
2. 流式会话槽精简（streaming_session.py）：SessionSlot 删除独立 req_pool_idx 字段与 is_holding_kv 属性；save_from_req 不再显式搬运 req.req_pool_idx 与置空（req.kv = ReqKvInfo() 重置时天然清掉），restore_to_req 不再回填（req.kv = copy.copy(self.kv) 已携带）。find_active_slot、release_session、session_held_tokens 等会话持有量统计读取点全部改用 slot.kv.is_held / slot.kv.req_pool_idx。
3. 分配器出入口与外围读者切换：ReqToTokenPool.alloc / free（memory_pool.py）与 DecodeReqToTokenPool.alloc / free（disaggregation/decode.py）的复用判断、写入、置空统一改走 r.kv.req_pool_idx；PD 分离侧 _mamba_payload、_swa_payload、

_swa_ring_payload、_c128_state_payload 等索引构建同步切换。scheduler.py 的 abort/chunked 处理、disaggregation/prefill.py 与decode_kvcache_offload_manager.py、hisparse_coordinator.py 的设备 / 主机缓冲与LRU 表、mem_cache/common.py 的 release_kv_cache /free_swa_out_of_window_slots、NPU dsv4 池、unified_radix_cache.py、mamba_radix_cache.py、sparse_coordinator.py、invariant_checker.py 等全部读者完成替换。

4. 测试配套：test_kv_page_invariants.py 将 _FakeReq 与 _FakeSlot 合并为 _FakeOwner，改用真实 ReqKvInfo 构造（不再手写 SimpleNamespace + is_holding_kv=True），使不变式检查器只依赖 kv 字段；test_streaming_session_unit.py、test_hisparse_unit.py 等 fixture 同步更新为 kv.req_pool_idx / kv.is_held 语义。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 核心结构；类别 source；类型 data-contract；符号 ReqKvInfo, Req, is_held, reset_for_retract）：ReqKvInfo 定义处，本次变更的核心：新增 req_pool_idx 字段与 is_held 属性，删除 Req.req_pool_idx / Req.is_holding_kv，所有读者的切换源头。
- python/sglang/srt/session/streaming_session.py（模块 流式会话；类别 source；类型 core-logic；符号 SessionSlot, save_from_req, restore_to_req, any_holding_kv）：SessionSlot 移除独立 req_pool_idx 字段与 is_holding_kv，save/restore 从逐字段搬运简化为整个 ReqKvInfo 拷贝，是本次迁移动机体现最充分的文件。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 alloc, free, set_mamba_ping_pong_slot, free_mamba_cache）：ReqToTokenPool.alloc/free 是行注册与注销的出入口，切换后所有分配者统一走 req.kv.req_pool_idx；mamba 映射表也按新路径取值。
- python/sglang/srt/disaggregation/decode.py（模块 PD 解码；类别 source；类型 core-logic；符号 DecodeReqToTokenPool.alloc, DecodeReqToTokenPool.free, pop_preallocated, _pre_alloc）：PD 分离 decode 侧所有 payload 构建（Mamba/SWA/ring/C128/hisparse）与预分配都读取 req_pool_idx，是最大的读者之一。
- python/sglang/srt/managers/hisparse_coordinator.py（模块 稀疏缓存；类别 source；类型 core-logic；符号 admit_request_into_staging, admit_request_direct, alloc_device_buffer, abort_staging_request）：HiSparse 协调器大量直接索引 req_pool_idx 管理 device/host buffer、staging 与 LRU 表，全面切换读取路径，验证低频路径的覆盖度。
- test/registered/unit/managers/test_kv_page_invariants.py（模块 页不变式；类别 test；类型 test-coverage；符号 _FakeOwner）：测试 fixture 的代表性重构：_FakeReq 与 _FakeSlot 合并为 _FakeOwner，改用真实 ReqKvInfo，验证不变式检查器不再依赖分散字段。

关键符号：ReqKvInfo.is_held, Req.reset_for_retract, Req.offload_kv_cache, Req.load_kv_cache, SessionSlot.save_from_req, SessionSlot.restore_to_req, StreamingSession.find_active_slot, StreamingSession.any_holding_kv, StreamingSession.release_session, ReqToTokenPool.alloc, ReqToTokenPool.free

关键源码片段

python/sglang/srt/managers/schedule_batch.py

ReqKvInfo 定义处，本次变更的核心：新增 req_pool_idx 字段与 is_held 属性，删除 Req.req_pool_idx / Req.is_holding_kv，所有读者的切换源头。

```
@dataclasses.dataclass(slots=True, kw_only=True)
class ReqKvInfo:
    # 设备 KV 的"所有权记录"：此前 req_pool_idx 散落在 Req 与 SessionSlot 上，
    # 现在统一收进本对象，持有多少 KV、注册在哪一行、是否已释放全部自包含。
    req_pool_idx: Optional[int] = None # req_to_token 行索引，即该记录的"注册行"

    # 请求自身的 KV 区间为 [cache_protected_len, kv_allocated_len)
    cache_protected_len: int = 0 # 前缀树拥有 [0, here)（已匹配或已插入）
    kv_committed_len: int = 0 # 已提交的 KV 内容长度，<= kv_allocated_len
    kv_allocated_len: int = 0

    # SWA 在 [swa_dead_lo(page_size), swa_evicted_seqlen) 的槽位已提前释放
    swa_evict_floor: int = 0 # [0, here) 从未被窗口驱逐（prefill 感知 SWA）
    swa_evicted_seqlen: int = 0 # SWA 驱逐游标

    @property
    def is_held(self) -> bool:
        # 存在性判定：有注册行即认为持有设备 KV，
        # 替代 Req.is_holding_kv 与 SessionSlot.is_holding_kv 两个包装器
        return self.req_pool_idx is not None

    @property
    def is_released(self) -> bool:
        return self.kv_allocated_len == 0 and self.swa_evicted_seqlen == 0

    def mark_released(self) -> None:
        self.kv_allocated_len = 0
        self.swa_evicted_seqlen = 0
```

python/sglang/srt/session/streaming_session.py

SessionSlot 移除独立 req_pool_idx 字段与 is_holding_kv，save/restore 从逐字段搬运简化为整个 ReqKvInfo 拷贝，是本次迁移动机体现最充分的文件。

```
def save_from_req(self, req: Req, is_first: bool):
    # 把即将结束的请求的 KV 状态存入槽位，完成所有权转移
    if is_first:
        # 只有首个请求把受保护前缀交给前缀树锁存，后续轮次不得再移动 KV
        self.last_node = req.last_node
        self.swa_uuid_for_lock = req.swa_uuid_for_lock
        self.skip_lock_node_ids = req.skip_lock_node_ids
    else:
        # 非首轮请求的受保护前缀必须与首次一致
        assert req.kv.cache_protected_len == self.kv.cache_protected_len
```

```
    # 整个 ReqKvInfo 拷贝即带走 req_pool_idx，无需再逐字段搬运
    self.kv = copy.copy(req.kv)

    # mamba 状态暂仍按字段搬运（尚未并入 ReqKvInfo，见后续 #37164 方向）
    self.mamba_pool_idx = req.mamba_pool_idx
    self.mamba_ping_pong_track_buffer = req.mamba_ping_pong_track_buffer
    self.mamba_next_track_idx = req.mamba_next_track_idx
    self.mamba_last_track_idx = req.mamba_last_track_idx
    self.mamba_last_track_seqlen = req.mamba_last_track_seqlen
    self.mamba_branching_seqlen = req.mamba_branching_seqlen

    # 清空请求侧引用，防止后续 alloc/retract 路径误用槽位所有的状态
    req.kv = ReqKvInfo()
    req.mamba_pool_idx = None
    req.mamba_ping_pong_track_buffer = None
    req.mamba_next_track_idx = None
    req.mamba_last_track_idx = None
    req.mamba_last_track_seqlen = None
    req.mamba_branching_seqlen = None


def restore_to_req(self, req: Req):
    # 把槽位 KV 状态恢复给进入的请求；kv 拷贝同样携带 req_pool_idx，无需单独回填
    req.kv = copy.copy(self.kv)
    req.swa_uuid_for_lock = self.swa_uuid_for_lock
    req.skip_lock_node_ids = self.skip_lock_node_ids

    req.mamba_pool_idx = self.mamba_pool_idx
    req.mamba_ping_pong_track_buffer = self.mamba_ping_pong_track_buffer
    req.mamba_next_track_idx = self.mamba_next_track_idx
    req.mamba_last_track_idx = self.mamba_last_track_idx
    req.mamba_last_track_seqlen = self.mamba_last_track_seqlen
    req.mamba_branching_seqlen = self.mamba_branching_seqlen

    # 注意：req_pool_idx 与 mamba_pool_idx 有意不从槽位清空——
    # chunked prefill 中请求可能被调度器拒绝并在下个周期重试，
    # 每次重试都会再次调用 restore_to_req，槽位必须保持幂等可恢复
```

评论区精华

本 PR 无任何 review 评论（comments_count 与 review_comments_count 均为 0），作者直接合入。能还原的设计思路来自 4 个 commit 的拆分：先做纯字段搬迁（move req_pool_idx into ReqKvInfo），再引入 is_held 并删除包装器（presence predicate on ReqKvInfo.is_held; drop is_holding_kv wrappers），最后统一测试假对象（merge kv page invariant fakes）。“先搬迁、后语义替换、再清理测试”的提交粒度让每个 commit 都可独立编译运行，是无行为变更的大型机械重构中值得借鉴的做法。另外 PR Test (Base) 与 AMD ROCm 7.2 两条 CI 记录为失败状态，但合入时未留下讨论记录，失败原因无法从本材料确认。

- 暂无高价值评论线程

风险与影响

- 风险：回归风险：68 个文件的 req.req_pool_idx → req.kv.req_pool_idx 机械替换可能遗漏 getattr / 反射访问点；不过 Req.req_pool_idx 是普通实例属性，删除后任何残留读点都会立刻 AttributeError 而非静默出错，这是本 PR 的天然防护。语义风险：mem_cache/common.py 的 release_kv_cache 断言 (not req.kv.is_held) == req.kv.is_released 依赖 req_pool_idx 与 allocated/swa 状态严格同步；而 mark_released 只清 kv_allocated_len / swa_evicted_seqlen 不清 req_pool_idx，与 save_from_req 的 req.kv = ReqKvInfo() 整体重置是两种"释放"语义，后续新增路径若只调 mark_released 不释放行会触发断言。CI 不确定性：PR Test (Base) 与 AMD ROCm 7.2 最后记录为失败，仅有 Extra 通过，失败原因未在材料中体现。兼容性：kv_canary perturb targets 与 MLX runner 的同名字段按 PR body 声明未动，但 unified_radix_cache.py、mamba_radix_cache.py、sparse_coordinator.py 等低关注度路径也在读者之列，需关注这些子系统的回归测试。
- 影响：系统层面：Req / SessionSlot 字段集变更，调度器、流式会话、PD 分离、HiSparse、NPU dsv4、unified/mamba radix cache 等所有读写点统一到 ReqKvInfo，设备 KV 的所有权信息（持有与否、持有区间、注册行、释放状态）收敛为单一事实源。团队层面：后续 KV ownership 相关开发以 ReqKvInfo 为唯一入口；测试中编造请求需构造真实 ReqKvInfo（如 _FakeOwner 模式）。用户层面：纯搬迁，无可见行为变化。
- 风险标记：核心路径数据结构变更，68 文件机械替换易遗漏，CI Base/ROCm 记录失败，无 review 独立审核

关联脉络

- PR #37164 [mem_cache] Move mamba state and retraction_backup into ReqKvInfo: 同属 kv-ownership 系列，把 mamba 状态与 retraction_backup 也并入 ReqKvInfo，与本 PR 方向一致、文件重叠（schedule_batch.py、memory_pool.py、streaming_session.py）。
- PR #37108 [mem_cache] Share one ReqKvInfo between a streaming session slot and its request: 同系列后继：流式会话槽与请求共享同一 ReqKvInfo，建立在本 PR 字段收敛之上。
- PR #37151 [Unified Cache Linker][3/N]: Add backend-independent linker core: 与 unified_radix_cache.py 同文件改动，同属 unified-radix-cache / memory-cache 领域，本 PR 也改动了该文件。
- PR #34602 feat(unified-memory): dense KV views for uniform-row MHA/SWA models: 统一内存池主线重构，使用同一 memory-pool 标签，后续与本 PR 的 ReqKvInfo 收敛共同构成 KV 所有权模型演进。