PR #36982 完整报告

sgl-project/sglang

[mem_cache] Move `cache_protected_len` and `swa_evict_floor` into `ReqKvInfo`

合并时间：2026-08-30 10:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36982

执行摘要

- 一句话：KV 记账字段收拢进 ReqKvInfo，纯重构为所有权模型铺路
- 推荐动作：值得精读。虽是纯重构，但展示了高质量 " 所有权集中化 " 迁移的完整范式：先收拢字段、再抽象计算逻辑（swa_dead_lo）、最后补显式不变式断言，且注释随 commit 反复打磨。对理解 sglang 的 KV cache 所有权模型、流式会话状态转移以及 unified-memory 演进方向都有帮助。重点阅读 schedule_batch.py 的 ReqKvInfo 定义与 streaming_session.py 的 save_from_req / try_match_prefix。

功能与动机

PR body 明确说明：需要把请求持有的 KV 记账（树拥有前缀终点、SWA 驱逐盾、驱逐游标、已分配长度）集中到一处，因为这些字段此前散落在 Req、SessionSlot、ReqKvInfo 多处，所有权转移（save_from_req / restore_to_req）只能逐字段搬运，容易漏同步；同时 SWA 驱逐下界 max(protected, shield) 的页对齐计算内联在 free_swa_out_of_window_slots 中，需要收敛。作者声明 "Pure field relocation with no behavior change; the session slot still pins the first request's protected prefix"，并注明 Follows up on #36958，分支名 lsyin/kv-ownership-pr2 表明这是所有权重构系列的第 2 个 PR。

实现拆解

1. 字段收拢（schedule_batch.py）：ReqKvInfo 新增 cache_protected_len 与 swa_evict_floor 字段，前者注释 "tree cache owns [0, here) (matched or inserted)"，后者注释 "[0, here) never window-evicted (prefill-aware SWA)"；Req.__init__不再单独初始化这两个属性，init_next_round_input 中 match_result 赋值为 self.kv.cache_protected_len，reset_for_retract 改为清零 self.kv.cache_protected_len（保留 swa_evict_floor，与旧行为一致）。
2. SWA 下界计算集中（schedule_batch.py + common.py）：新增方法 ReqKvInfo.swa_dead_lo(page_size)，封装 lo = max(cache_protected_len, swa_evict_floor)，且仅当 lo > cache_protected_len 时用 ceil_align 向上页对齐；common.py 的 free_swa_out_of_window_slots 删除内联的 evict_floor 计算（含 getattr(req, "swa_evict_floor", 0) 防御访问），改为 req.kv.swa_evicted_seqlen = max(..., req.kv.swa_dead_lo(page_size))。旧式 -(-x // page_size) * page_size 与 ceil_align 语义等价。

3. 访问路径迁移（radix 系缓存）：radix_cache.py、swa_radix_cache.py、mamba_radix_cache.py、pure_swa_radix_cache.py、chunk_cache.py、unified_radix_cache.py 的 cache_finished_req / cache_unfinished_req 中全部 req.cache_protected_len 改为 req.kv.cache_protected_len，覆盖 free_segment 边界、InsertParams.prev_prefix_len、assert 表达与 req_to_token_pool.write 切片。
4. SessionSlot 去重并补断言（streaming_session.py）：删除 SessionSlot.cache_protected_len 独立字段，try_match_prefix 的 NPU 对齐检查、assert prefix_len >= slot.kv.cache_protected_len、MatchResult.cache_protected_len、release_session 的 protected_len、session_held_tokens / session_held_swa_tokens 统计全部改读 slot.kv.cache_protected_len；save_from_req 在 is_first=False 分支新增断言，固化 " 受保护前缀是首轮请求的树锁，后续轮次不得移动 " 的不变式，同时将 self.kv = copy.copy(req.kv) 移到 if/else 之后保证整行记账一并转移。
5. 外围模块同步：disaggregation/common/staging_handler.py（decode 前缀页对齐校验与 _scatter_region 的 prefix_tokens）、disaggregation/decode.py、managers/scheduler.py、schedule_policy.py、scheduler_components/invariant_checker.py（full_uncached / swa_uncached 统计）同步字段路径。
6. 测试与文档配套：约 14 个测试文件同步适配，代表性包括 test_unified_radix_cache_unittest.py、test_streaming_session_unit.py、test_pure_swa_chunk_cache.py、test_swa_unittest.py、test_swa_eviction_boundary.py、test_scheduler_chunked_req_gate.py；chunk_cache.py 与 streaming_session.py 的注释同步更新为 req.kv.swa_evict_floor / slot.kv.cache_protected_len 引用。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 调度批处理；类别 source；类型 core-logic；符号 ReqKvInfo.swa_dead_lo, ReqKvInfo.is_released, ReqKvInfo.mark_released, Req.init_next_round_input）：重构核心：ReqKvInfo 新增 cache_protected_len、swa_evict_floor 字段与 swa_dead_lo 方法，Req.init/init_next_round_input / reset_for_retract 的字段路径全部迁移，是全部 29 个文件改动的源头。
- 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, SessionSlot.restore_to_req, SessionSlot.try_match_prefix, SessionSlot.release_session）：SessionSlot 删除独立 cache_protected_len 字段并统一改读 slot.kv；save_from_req 新增非首轮断言，是本 PR 唯一的行为性变更与关键不变式。
- python/sglang/srt/mem_cache/common.py（模块 SWA 驱逐；类别 source；类型 core-logic；符号 free_swa_out_of_window_slots）：free_swa_out_of_window_slots 删除内联的 evict_floor 页对齐计算，改用 ReqKvInfo.swa_dead_lo，验证了集中化抽象的正确性。
- python/sglang/srt/mem_cache/mamba_radix_cache.py（模块 Mamba 缓存；类别 source；类型 core-logic；符号 MambaRadixCache.cache_finished_req, MambaRadixCache.cache_unfinished_req）：Mamba 缓存的 cache_finished_req /

cache_unfinished_req 中受保护前缀字段路径全部迁移，是覆盖面最大的机械替换文件之一。

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic；符号 UnifiedRadixCache.cache_finished_req, UnifiedRadixCache.cache_unfinished_req）：统一缓存（含 SWA 窗口外释放路径）的插入与释放逻辑同步迁移，验证字段收拢对组合缓存架构同样成立。
- python/sglang/srt/mem_cache/radix_cache.py（模块 Radix 缓存；类别 source；类型 core-logic；符号 RadixCache.cache_finished_req, RadixCache.cache_unfinished_req）：经典 radix cache 的 cache_finished_req / cache_unfinished_req 字段路径迁移，注释点明 cache_protected_len 与 prefix_indices 长度在 page_size > 1 时的差异。
- python/sglang/srt/mem_cache/chunk_cache.py（模块 分块缓存；类别 source；类型 core-logic；符号 SWAChunkCache.cache_finished_req, PureSWAChunkCache.cache_finished_req）：SWAChunkCache / PureSWAChunkCache 的完成请求释放逻辑同步迁移，并更新 swa_evict_floor 相关注释。
- python/sglang/srt/disaggregation/common/staging_handler.py（模块 PD 中转；类别 source；类型 entrypoint；符号 StagingHandler.register_decode_req, StagingHandler._scatter_region）：PD 中转入口的 decode 前缀页对齐校验与 scatter 偏移计算同步迁移，说明字段收拢影响面延伸至 disaggregation 路径。
- python/sglang/srt/managers/scheduler_components/invariant_checker.py（模块 不变式检查；类别 source；类型 core-logic；符号 InvariantChecker._get_total_uncached_sizes）：调度不变式检查器的未缓存统计（full / swa）同步字段路径，保持与生产逻辑一致。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage）：统一缓存单元测试全量适配新字段路径，是本 PR 覆盖面最大的测试改动（+14/-14）。
- test/registered/unit/mem_cache/test_streaming_session_unit.py（模块 缓存测试；类别 test；类型 test-coverage）：流式会话单元测试同步适配 slot.kv 路径，覆盖新增断言相关的多轮保存 / 恢复场景。

关键符号：ReqKvInfo.swa_dead_lo, ReqKvInfo.is_released, ReqKvInfo.mark_released, Req.init_next_round_input, Req.reset_for_retract, SessionSlot.save_from_req, SessionSlot.restore_to_req, SessionSlot.try_match_prefix, SessionSlot.release_session, free_swa_out_of_window_slots, RadixCache.cache_finished_req, UnifiedRadixCache.cache_unfinished_req, MambaRadixCache.cache_unfinished_req, PureSWAChunkCache.cache_finished_req, StagingHandler._scatter_region

关键源码片段

python/sglang/srt/managers/schedule_batch.py

重构核心：ReqKvInfo 新增 cache_protected_len、swa_evict_floor 字段与 swa_dead_lo 方法，Req.init/ init_next_round_input / reset_for_retract 的字段路径全部迁移，是全部 29 个文件改动的源头。

```
@dataclasses.dataclass(slots=True, kw_only=True)
class ReqKvInfo:
```

```
    # 请求在 prefix cache 之外持有的设备 KV 记账，是请求 KV 所有权的单一载体。
    # 每个 Req 上始终存在；是否真正持有 KV 由 req.req_pool_idx is not None 决定。

    # 请求自身拥有的 KV 区间为 [cache_protected_len, kv_allocated_len)。
    cache_protected_len: int = 0 # 树缓存拥有 [0, here)（命中或插入的部分）
    kv_allocated_len: int = 0

    # SWA 区间 [swa_dead_lo(page_size), swa_evicted_seqlen) 已被手动释放。
    swa_evict_floor: int = 0 # [0, here) 永不被窗口驱逐（prefill 感知型 SWA）
    swa_evicted_seqlen: int = 0 # SWA 驱逐游标

    def swa_dead_lo(self, page_size: int) -> int:
        # 本请求可自行释放的最低 SWA 位置：必须在树拥有前缀与驱逐盾之上，
        # 且仅在确实高于 cache_protected_len 时向上按页对齐。
        lo = max(self.cache_protected_len, self.swa_evict_floor)
        if page_size > 1 and lo > self.cache_protected_len:
            lo = ceil_align(lo, page_size)
        return lo

    @property
    def is_released(self) -> bool:
        # 已分配长度与驱逐游标都为 0 时视为已释放。
        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 删除独立 cache_protected_len 字段并统一改读 slot.kv；save_from_req 新增非首轮断言，是本 PR 唯一的行为性变更与关键不变式。

```
  def save_from_req(self, req: Req, is_first: bool):
        """把一个即将结束的请求的 KV 状态保存进本 slot。"""
        self.req_pool_idx = req.req_pool_idx
        self.kv_committed_len = req.kv_committed_len

        if is_first:
            # 首轮才记录树锁信息：受保护前缀是首轮请求持有的树锁，
            # 后续轮次不再向树移交 KV，因此该值必须保持不变。
            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:
            # 显式不变式：非首轮请求的受保护前缀必须与 slot 已保存的一致，
            # 否则说明某个路径移动了树锁边界，属于编程错误。
            assert req.kv.cache_protected_len == self.kv.cache_protected_len

        # 整行转移 KV 记账的所有权：ReqKvInfo 是纯标量字段，浅拷贝即安全。
```

```
        self.kv = copy.copy(req.kv)

        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

        # 所有权已转移到 slot：清空 req 上的引用，避免后续 alloc / retract 路径
        # 把 slot 持有的张量误当成请求自身的资源。
        req.req_pool_idx = None
        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
```

python/sglang/srt/mem_cache/common.py

free_swa_out_of_window_slots 删除内联的 evict_floor 页对齐计算，改用 ReqKvInfo.swa_dead_lo，验证了集中化抽象的正确性。

```
  def free_swa_out_of_window_slots(
        req: Req,
        pre_len: int,
        *,
        sliding_window_size: int,
        page_size: int,
        req_to_token_pool: ReqToTokenPool,
        token_to_kv_pool_allocator: BaseTokenToKVPoolAllocator,
        is_chunk_cache: bool = False,
        retain_floor: int | None = None,
  ) -> None:
        if not req.is_holding_kv:
            return

        # SWA radix cache 场景：驱逐既不在树缓存里、又已滑出窗口的 token。
        assert (
            req.kv.cache_protected_len % page_size == 0
        ), "cache_protected_len must be page aligned"

        # 驱逐游标至少推进到 swa_dead_lo：低于它的区间归树 / 驱逐盾所有，
        # 请求无权释放；对齐逻辑统一收敛在 ReqKvInfo.swa_dead_lo 中。
        req.kv.swa_evicted_seqlen = max(
            req.kv.swa_evicted_seqlen, req.kv.swa_dead_lo(page_size)
        )
```

```
    if is_chunk_cache:
        # Chunk cache 不建 radix 树，没有 tombstone 叶节点问题，驱逐到窗口边界即可。
        evict_threshold = pre_len - sliding_window_size
    else:
        # Radix cache：至少保留 max(window, page)，并让前沿低于插入边界
        #  （page_floor(seq_len)），保证最后一个叶节点不会是全 tombstone。
        evict_threshold = pre_len - max(sliding_window_size, page_size)

    # retain_floor 由调用方（BasePrefixCache.swa_retain_floor）决定，
    # 这里只承诺不越过它释放；chunk cache 无树可共享，保留无收益。
    if retain_floor is not None and not is_chunk_cache:
        evict_threshold = min(evict_threshold, retain_floor)

    new_swa_evicted_seqlen = max(req.kv.swa_evicted_seqlen, evict_threshold)
    if page_size > 1:
        new_swa_evicted_seqlen = (new_swa_evicted_seqlen // page_size) * page_size

    if new_swa_evicted_seqlen > req.kv.swa_evicted_seqlen:
        free_slots = req_to_token_pool.req_to_token[
            req.req_pool_idx, req.kv.swa_evicted_seqlen : new_swa_evicted_seqlen
        ]
        token_to_kv_pool_allocator.free_swa(free_slots)
        req.kv.swa_evicted_seqlen = new_swa_evicted_seqlen
```

评论区精华

本 PR 没有 Review 评论（comments_count 与 review_comments_count 均为 0），设计考量体现在 6 个 commit 的演进中：① 首个 commit 一次性完成字段迁移与 swa_dead_lo 引入；② 随后两个 commit 反复整理 ReqKvInfo 的字段布局与内联注释（tidy layout / inline field comments）；③ 第 4 个 commit 新增 session 断言，是唯一的行为性提交，把 " 首轮后受保护前缀不变 " 的隐式契约显式化；④ 第 5 个 commit 补充注释后合并 main。整体呈现 " 先机械迁移、再抽象收敛、最后补不变式断言 " 的清晰重构节奏。

- SessionSlot 是否保留独立 cache_protected_len 字段 (design): 删除 SessionSlot.cache_protected_len 独立字段，统一读取 slot.kv.cache_protected_len，并在 save_from_req 非首轮分支增加等式断言。
- swa_dead_lo 抽象是否等价于旧内联计算 (correctness): 逐行对照语义等价（-(-x // page_size) * page_size 即 ceil_align），测试同步覆盖驱逐边界，确认无行为变化。
- 注释与布局反复打磨的提交节奏 (style): 注释定稿，行为保持纯迁移；这不是行为争议，而是对热路径记账字段可读性的工程投入。

风险与影响

- 风险：
  1. 机械替换遗漏风险：29 个文件中上百处 req.cache_protected_len → req.kv.cache_protected_len，任何遗漏都会在运行期抛 AttributeError；依赖约 14 个测试文件覆盖，尤其是 test_unified_radix_cache_unittest.py 与

test_streaming_session_unit.py。

2. 新增断言风险：流式会话多轮场景首次生效的 assert req.kv.cache_protected_len == self.kv.cache_protected_len，若存在某条路径在后续轮次推进受保护前缀（如 retract 后 prefix 被重新匹配），会在生产环境触发；该断言语义上依赖 session_controller 的 append-only 保证（代码注释已说明）。
3. 字段双副本的中间态：本 PR 之后 Req 与 SessionSlot 仍各持有一个 ReqKvInfo 浅拷贝（copy.copy），mark_released 不清 cache_protected_len / swa_evict_floor 的行为保持不变，但这是系列重构中间态，后续 #37108 将共享同一对象，中途需要保证两副本语义一致。
4. 热路径变更：schedule_batch.py、radix 系缓存与调度器统计路径均为每请求高频路径，字段访问从 Req 直接属性变为 self.kv 间接访问，虽为 Python 层纯属性读取、开销可忽略，但回归影响面广。 - 影响：用户层面无任何 API 或行为变化（纯字段迁移）；系统层面，SWA 驱逐下界计算从 free_swa_out_of_window_slots 内联逻辑收敛为 ReqKvInfo.swa_dead_lo 单一实现点，消除了三处重复语义；流式会话的所有权转移改为整行 ReqKvInfo 复制并新增不变式保护。团队层面，本 PR 是 kv-ownership 系列的关键中间步骤：ReqKvInfo 由此成为请求持有 KV 记账的单一载体，直接支撑后续 #37094（req_pool_idx）、#37108（slot 与请求共享 ReqKvInfo）、#37164（mamba 状态收拢）的推进。 - 风险标记：跨 29 文件机械迁移，调度热路径字段路径变更，新增 session 轮次不变式断言，Req 与 slot 字段双副本中间态，依赖测试覆盖防遗漏

关联脉络

- PR #37094 [mem_cache] Move req_pool_idx into ReqKvInfo: 同一条 kv-ownership 重构线的直接下一步：把 req_pool_idx 与 is_held 也收进 ReqKvInfo，本 PR 的字段收拢是其前置条件。
- PR #37108 [mem_cache] Share one ReqKvInfo between a streaming session slot and its request: 本 PR 的后续：让流式会话 slot 与请求共享同一个 ReqKvInfo 对象，消除 save_from_req / restore_to_req 的 copy.copy 副本，正是本 PR 集中字段后才有意义的改造。
- PR #37164 [mem_cache] Move mamba state and retraction_backup into ReqKvInfo: 系列继续推进，把 mamba 状态与 retraction_backup 也收进 ReqKvInfo，实现请求 KV 所有权状态的真正统一。
- PR #37182 [CI] Fix unreachable FakeReq field initialization: 修复 test_streaming_session_unit.py 中 FakeReq 的不可达初始化并补 Mamba 状态回归测试，与本 PR 改动的流式会话测试同文件相关联。
- PR #35245 refactor(unified-memory): translate the KV write location once, at ForwardBatch construction: unified-memory 线路上的同类字段集中化重构，与 kv-ownership 系列共同体现 KV 记账与索引集中管理的演进方向。