PR #37078 完整报告

sgl-project/sglang

[mem_cache] Move `kv_committed_len` into `ReqKvInfo`

合并时间：2026-08-30 13:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37078

执行摘要

- 一句话：将 kv_committed_len 收进 ReqKvInfo，统一 KV 行三段长度记录
- 推荐动作：值得快速浏览并作为理解 SGLang KV 所有权模型的起点：ReqKvInfo 的三段式长度阶梯是后续所有 KV 生命周期逻辑（分配、提交、保护、SWA 淘汰）的统一入口。对内存管理模块感兴趣的同学建议精读 schedule_batch.py 的 ReqKvInfo 定义与 streaming_session.py 的 save_from_req / restore_to_req 模式，它展示了如何用『一条可整体拷贝的记录』替代『多处散落字段的手工同步』。

功能与动机

PR body 明确说明动机：Move kv_committed_len from Req / SessionSlot into ReqKvInfo, next to cache_protected_len and kv_allocated_len, so the whole [protected, committed, allocated] ladder of a request's KV row is one record. 此前一个请求的 KV 生命周期由散落在多处、且 Req 与 SessionSlot 各持一份的字段描述，save/restore 需要逐字段手工同步，漏掉任何一个字段都会造成 committed/allocated 不一致。收拢后，整个 KV 行的状态可以随一条可整体拷贝的记录流转，这是后续 KV 所有权统一重构（req_pool_idx、Mamba 状态等陆续并入 ReqKvInfo）的基础。

实现拆解

1. 字段定义收拢（schedule_batch.py）：在 ReqKvInfo dataclass 中新增 kv_committed_len: int = 0，位置夹在 cache_protected_len 与 kv_allocated_len 之间，注释明确 committed <= allocated 的约束；同时删除 Req.__init__ 里的 self.kv_committed_len = 0，从此该状态只有一处定义。
2. Req 生命周期读写点迁移（schedule_batch.py）：effective_kv_committed_len（strip_thinking_cache 时用 min(kv.kv_committed_len, len(origin_input_ids)) 截断）、reset_for_retract、prepare_for_extend（保留 TODO(th4) 不动）、prepare_for_decode（每个 decode 步骤提交 +1）、new_tokens_required_next_decode 及其 spec-v2 版本、mamba_lazy_spec_in_window 全部改为读写 req.kv.kv_committed_len。
3. SessionSlot 简化（streaming_session.py）：删除 SessionSlot 上独立的 kv_committed_len 字段；save_from_req / restore_to_req 不再手工复制该字段，改由整份 self.kv = copy.copy(req.kv) 携带。NPU page 对齐分支、_free_tail、_trim_overshoot 等同步改读写 slot.kv.kv_committed_len / req.kv.kv_committed_len。同时保留并强化了 cache_protected_len 在首轮后不可变的断言。

4. 下游子系统同步迁移：batch_result_processor.py 的 spec-v2 接受长度累加与 Mamba 边界检测（_resolve_spec_v2_tokens、_mamba_prefix_cache_update、_mamba_check_track_boundary、_mamba_assert_committed_len_lookahead）、disaggregation/decode.py 的 _pre_alloc 与 get_new_prebuilt_batch、invariant_checker.py 的 _check_kv_page_invariants（守护 committed<=allocated 不变量）、beam_search 的 fork.py 与 coordinator.py（leader 行释放与幸存者同步）、NPU dsv4_allocator.py、flexkv/lmcache 后端、dflash_info_v2.py、allocation_sizing.py 等全部改为 kv.kv_committed_len。
5. 测试与不变量配套：test_streaming_session_unit.py 的 _FakeReq 把 committed 从平铺属性移入 kv 子对象，SessionSlot(...) 构造同步调整；test_unified_radix_cache_unittest.py 有 11 处 req.kv_committed_len = kv_len 改为 req.kv.kv_committed_len = kv_len；test_decode_radix_lock_ref.py、test_dllm_fdfo_kv_reuse.py、test_kv_page_invariants.py、test_fork.py 等也同步更新。PR 分支在过程中两次 merge main 解决与主线的冲突。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 请求调度；类别 source；类型 core-logic；符号 ReqKvInfo, Req, effective_kv_committed_len, reset_for_retract）：本 PR 的核心定义处：在 ReqKvInfo 中新增 kv_committed_len 字段，删除 Req 上的平级属性，并迁移 effective_kv_committed_len、reset_for_retract、prepare_for_extend、prepare_for_decode、mamba 窗口判断与页数估算等调度主路径的读写。
- 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, try_match_prefix）：SessionSlot 删除独立 kv_committed_len 字段，save_from_req / restore_to_req 不再逐字段复制而由整份 ReqKvInfo 拷贝携带；NPU page 对齐、_free_tail、_trim_overshoot 等同步迁移，是本次简化收益最明显的文件。
- python/sglang/srt/managers/scheduler_components/batch_result_processor.py（模块 结果处理；类别 source；类型 core-logic；符号 _resolve_spec_v2_tokens, _handle_finish_state_updated_req, _mamba_prefix_cache_update, _mamba_check_track_boundary）：spec-v2 接受 token 提交（kv_committed_len += num_accept_tokens）与 Mamba 边界检测、lookahead 断言全部改读 req.kv.kv_committed_len，是 decode 阶段提交语义的关键读写点。
- python/sglang/srt/disaggregation/decode.py（模块 解码侧；类别 source；类型 core-logic；符号 alloc, _pre_alloc, get_new_prebuilt_batch）：PD 解码预分配路径 _pre_alloc 设置 req.kv.kv_committed_len 并用于 set_extend_range 截断；alloc 复用断言与 get_new_prebuilt_batch 同步迁移。
- python/sglang/srt/managers/scheduler_components/invariant_checker.py（模块 不变量检查；类别 source；类型 core-logic；符号 _check_kv_page_invariants, _add_owner）：KV 页不变量检查（committed <= allocated 且无双重释放）同步读取 req/slot 的 kv.kv_committed_len，保证迁移后不变量守护逻辑仍然有效。
- test/registered/unit/mem_cache/test_streaming_session_unit.py（模块 会话测试；类别 test；类型 test-coverage；符号 _FakeReq, test_preabort_detaches_session_and_prese

rves_slot, test_nth_mid_abort_nukes_session_slot,test_release_session_threads_mamba_skip_ids）：测试 fake（_FakeReq / SessionSlot构造）从平铺 kv_committed_len 改为 kv 子对象承载，直接验证字段删除后 save/restore行为的正确性。

- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage；符号 _make_req, test_cache_finished_req_insert, test_cache_unfinished_req）：覆盖 radix cache 插入、SWA 淘汰等路径的 11 处 committed 设置全部改写 req.kv.kv_committed_len，防止核心缓存路径回归。
- python/sglang/srt/beam_search/fork.py（模块 束搜索；类别 source；类型 core-logic；符号 free_member_rows）：beam search 行释放时 leader 的 kv_committed_len 与 kv_allocated_len 需要一致重置，迁移后改为一并在 kv 记录内更新。

关键符号：effective_kv_committed_len, reset_for_retract, prepare_for_extend, prepare_for_decode, save_from_req, restore_to_req, try_match_prefix, _resolve_spec_v2_tokens, _mamba_check_track_boundary, _pre_alloc, _check_kv_page_invariants, free_member_rows

关键源码片段

python/sglang/srt/managers/schedule_batch.py

本 PR 的核心定义处：在 ReqKvInfo 中新增 kv_committed_len 字段，删除 Req 上的平级属性，并迁移 effective_kv_committed_len、reset_for_retract、prepare_for_extend、prepare_for_decode、mamba 窗口判断与页数估算等调度主路径的读写。

```
@dataclasses.dataclass(slots=True, kw_only=True)
class ReqKvInfo:
    # 设备侧 KV 行状态：请求在前缀缓存之外持有的 KV。
    # 是否持有以 req.req_pool_idx is not None 为准（Req.is_holding_kv）。

    # 三段式长度阶梯，描述一行 KV 的生命周期：
    # [0, cache_protected_len) 树缓存拥有（匹配或插入），不可回收
    # [cache_protected_len, kv_committed_len) 内容已提交，可进入前缀缓存
    # [kv_committed_len, kv_allocated_len) 已分配但未提交，可回滚
    cache_protected_len: int = 0 # 树缓存拥有 [0, here)（匹配或插入）
    kv_committed_len: int = 0 # KV 内容提交到 here，满足 committed <= allocated
    kv_allocated_len: int = 0

    # SWA 窗口： [swa_dead_lo(page_size), swa_evicted_seqlen) 已经释放
    swa_evict_floor: int = 0 # [0, here) 永远不会被窗口淘汰（prefill 感知 SWA）
    swa_evicted_seqlen: int = 0 # SWA 淘汰游标

    def swa_dead_lo(self, page_size: int) -> int:
        # 请求可以自行释放的最低 SWA 位置：高于树拥有前缀与淘汰保护线，
        # 并向上按页对齐。
        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:
        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 kv_committed_len 字段，save_from_req / restore_to_req 不再逐字段复制而由整份 ReqKvInfo 拷贝携带；NPU page 对齐、_free_tail、_trim_overshoot 等同步迁移，是本次简化收益最明显的文件。

```
@dataclass
class SessionSlot:
    """Holds KV state between streaming session turns."""

    virtual_node: _VirtualNode = field(default_factory=_VirtualNode)

    # KV 池状态：整行 ReqKvInfo 随会话保存，不再单独维护 kv_committed_len
    req_pool_idx: Optional[int] = None
    kv: ReqKvInfo = field(default_factory=ReqKvInfo)
    # ... 其余 mamba 状态字段保持不变

    def save_from_req(self, req: Req, is_first: bool):
        """将一个结束请求的 KV 状态保存进 slot。"""
        self.req_pool_idx = req.req_pool_idx

        if is_first:
            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:
            # 受保护前缀来自第一个请求的树锁；此后没有新 KV 交给树，
            # 所以后续轮次不得移动它，这里用断言固定该契约。
            assert req.kv.cache_protected_len == self.kv.cache_protected_len

        # 整行 KV 记录随 copy.copy 转移；kv_committed_len 不再单独复制，
        # 避免 Req 与 slot 各持一份导致的手工同步遗漏。
        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

        # 所有权移交给 slot；清空 req 的引用，避免后续分配 / 回退路径
```

```
        # 把 slot 持有的 mamba 状态误认为自己的。
        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
    # restore_to_req 同理：req.kv = copy.copy(self.kv) 一次带回全部字段
```

评论区精华

该 PR 没有任何 GitHub review 评论（comments_count 与 review_comments_count 均为 0），设计约束主要通过 commit message 与代码注释沉淀：

- 前序 commit "move cache_protected_len, swa_evict_floor into ReqKvInfo; add swa_dead_lo" 说明这是分步重构，先收内存保护与 SWA 字段，再收 committed 字段。
- commit "assert session cp fixed after first turn" 把『流式会话首轮之后，受保护前缀不可再移动』固化为 save_from_req 中的断言，防止后续轮次破坏树锁契约。
- commit "drop dead kv_committed_len attrs from test fakes" 说明迁移完成后，测试假对象上的死属性被一并清理，避免新旧两套字段长期共存造成测试失真。
- 暂无高价值评论线程

风险与影响

- 风险：
  1. 机械替换遗漏风险：37 个文件的批量替换依赖替换完整性与测试覆盖。任何遗漏的直接访问 req.kv_committed_len / slot.kv_committed_len 会立即触发 AttributeError（字段已删除），属于『快速失败』型错误；但若遗漏点只在特定硬件或特定路径（如 NPU、flexkv/lmcache 后端、beam search 分支）出现，则可能要到运行期才暴露。
  2. SessionSlot 字段删除影响面：streaming_session.py 删除了 SessionSlot.kv_committed_len，所有流式会话外的代码如果直接构造或读取该字段会编译期不可见、运行期报错；save_from_req / restore_to_req 的行为正确性现在完全依赖 copy.copy(req.kv) 携带全部字段。
  3. 不变量守护依赖：committed <= allocated 的约束仍由 invariant_checker.py 的 _check_kv_page_invariants 守护，该检查同时覆盖 req 与 slot，迁移后两者都从 kv.kv_committed_len 读取，守护逻辑本身需要保持正确。
  4. 行为敏感点：NPU page 对齐分支（try_match_prefix 中对 req.kv_committed_len 与 slot.kv_committed_len 同时 min 截断）、mamba lookahead 断言（kv_committed_len - token_seq_len in (0, 1)）以及 beam search fork.py 中 leader.kv_committed_len = start 与 kv_allocated_len = start 的联合重置，都属于改动等价但语义敏感的位置。 - 影响：对用户无任何可感知的功能变化；对系统而言，KV 行的三段长度（protected/committed/allocated）首次集中在同一条 ReqKvInfo 记录中，消除了 Req 与 SessionSlot 之间逐字段拷贝的一致性隐患。对团队而言，这是 KV 所

有权统一重构的中间步骤，后续修改 KV 行状态只需关注 ReqKvInfo 一处定义；同时为 streaming session 的 slot/request 共享同一条记录（PR #37108）铺平了道路。影响面广但风险受控，属于『大范围、低风险、高一致性收益』的重构。 - 风险标记：37 文件跨子系统机械替换，核心路径字段迁移，SessionSlot 字段删除影响面广，依赖测试覆盖验证无回归

关联脉络

- PR #37094 [mem_cache] Move req_pool_idx into ReqKvInfo: 与本 PR 同属 KV 所有权统一系列，将请求池索引也收进 ReqKvInfo，并引入 is_held 概念，属于同一演进方向的后续重构。
- PR #37108 [mem_cache] Share one ReqKvInfo between a streaming session slot and its request: 在 kv_committed_len 等字段收拢进 ReqKvInfo 的基础上，让流式会话 slot 与请求共享同一条 ReqKvInfo 对象，直接受益于本 PR 的整行记录统一。
- PR #37164 [mem_cache] Move mamba state and retraction_backup into ReqKvInfo: 继续把 Mamba 状态与回退备份收进 ReqKvInfo，是同一系列中进一步充实 KV 行记录内容的后续 PR。