PR #36958 完整报告

sgl-project/sglang

[mem_cache] Keep `req.kv` non-optional and key KV ownership on `req_pool_idx`

合并时间：2026-08-29 14:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36958

执行摘要

- 一句话：Req.kv 常驻化，KV 归属统一由 req_pool_idx 判定
- 推荐动作：值得精读。这是 SGLang mem_cache KV 所有权重构系列的开篇（分支名 lsyin/kv-ownership-pr1），is_holding_kv 单一谓词 + eager 对象 + 清零释放的组合是后续 #37094、#37108、#37164 的阅读前提。建议重点理解三处：ReqKvInfo.is_released/mark_released 的契约、release_kv_cache 收紧后的断言、以及 streaming_session.py 中 slot 与 req 之间的所有权转移方式。

功能与动机

PR body 明确指出：Keep req.kv on the Req for its whole lifetime (zeroed via mark_released() instead of set to None) and make req_pool_idx is not None the single "holds KV" predicate。row 与 KV 簿记本来就同时分配、同时释放，三个 if req.kv is None 分配分支说明对象创建被分散在 allocation.py、decode.py 等多处，eager 化后可以全部折叠，并把生命周期断言从两个平行 Optional 状态收敛为互斥一致的不变量。

实现拆解

1. ReqKvInfo eager 化与生命周期方法：在 python/sglang/srt/managers/schedule_batch.py 中，ReqKvInfo 两个字段（kv_allocated_len、swa_evicted_seqlen）获得默认值 0，新增 is_released 属性（两字段同时为 0）与 mark_released() 方法（清零两字段）；Req.__init__ 中 self.kv 由 Optional[ReqKvInfo] = None 改为 ReqKvInfo()；新增 Req.is_holding_kv 属性（return self.req_pool_idx is not None）；reset_for_retract 中断言由 assert self.kv is None 改为 assert not self.is_holding_kv。
2. 谓词替换：所有 req.kv is None 判断替换为 req.is_holding_kv 或 not req.is_holding_kv。涉及：mem_cache/common.py 的 free_swa_out_of_window_slots 与 release_kv_cache、schedule_batch.py 的 maybe_evict_swa、managers/scheduler_components/invariant_checker.py 的 _get_total_uncached_sizes 与 _add_owner、disaggregation/prefill.py 的 handle_bootstrap_failure、managers/scheduler_pp_mixin.py 的 PP 动态块 release 路径。
3. 折叠分配分支：mem_cache/allocation.py 的 alloc_for_extend 与 _alloc_extend_loc_with_kv_reuse、disaggregation/decode.py 的 alloc_for_decode_prealloc_hisparse 与 alloc_for_decode_prealloc 中，三处 if req.kv is None: req.kv = ReqKvInfo(...) 分支全部删除，变为直接赋值 req.kv.kv_allocated_len

= fill_len / seq_len，同时移除相应局部 ReqKvInfo 导入。

4. 释放路径统一 mark_released()：mem_cache/common.py 的 release_kv_cache 中，生命周期断言由 (req.req_pool_idx is None) == (req.kv is None) 收紧为 (not req.is_holding_kv) == req.kv.is_released，末尾 req.kv = None 改为 req.kv.mark_released()；managers/scheduler_pp_mixin.py、disaggregation/decode_kvcache_offload_manager.py 的 _release_finished_req 同样改用 mark_released()；session/streaming_session.py 的 save_from_req 与 try_cache_finished_req 在所有权转移给 slot 时将 req.kv = None 改为 req.kv = ReqKvInfo()，同时 SessionSlot.kv 改为 field(default_factory=ReqKvInfo)、SessionSlot.is_holding_kv 改为基于 req_pool_idx。
5. 测试与辅助适配：8 个测试文件（test_unified_radix_cache_unittest.py、test_dllm_fdfo_kv_reuse.py、test_specv2_kvcache_offloading.py、test_unified_radix_cache_bench.py、test_hisparse_allocator.py、test_kv_page_invariants.py、test_swa_eviction_boundary.py，以及 python/sglang/test/scripted_runtime/req_handle.py）适配 *eager* kv 语义，移除对 kv=None 的假设。配套变更的关键在于：断言从“两个 None 平行”转为“谓词与清零互斥一致”，任何遗漏 mark_released() 的路径都会被立即暴露。

关键文件：

- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic；符号 is_released, mark_released, is_holding_kv, ReqKvInfo）：核心数据模型变更入口：ReqKvInfo 常驻化、is_holding_kv 单一谓词、断言收紧都在此定义
- python/sglang/srt/mem_cache/common.py（模块 缓存管理；类别 source；类型 core-logic；符号 release_kv_cache, free_swa_out_of_window_slots, mark_released）：所有请求结束时的 KV 释放主路径，生命周期不变式与 mark_released 语义在此落地
- python/sglang/srt/session/streaming_session.py（模块 流式会话；类别 source；类型 dependency-wiring；符号 SessionSlot, save_from_req, find_active_slot, is_holding_kv）：流式会话 slot 与 req 间 KV 所有权转移的重构关键，slot 状态与 req 状态同步调整
- python/sglang/srt/mem_cache/allocation.py（模块 KV 分配；类别 source；类型 dependency-wiring；符号 alloc_for_extend, _alloc_extend_loc_with_kv_reuse）：展示分支折叠收益：三处 kv 空判断中最典型的分配入口
- python/sglang/srt/disaggregation/decode.py（模块 PD 分离；类别 source；类型 core-logic；符号 alloc_for_decode_prealloc, alloc_for_decode_prealloc_hisparse）：PD 分离场景下 decode 预分配路径的分支折叠与 ReqKvInfo 导入清理
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 单元测试；类别 test；类型 test-coverage）：变更覆盖最广的测试适配文件，验证 *eager* kv 语义下统一 radix cache 行为不变

关键符号：ReqKvInfo.is_released, ReqKvInfo.mark_released, Req.is_holding_kv, Req.reset_for_retract, SessionSlot.is_holding_kv, SessionSlot.save_from_req, release_kv_cache, free_swa_out_of_window_slots, alloc_for_decode_prealloc, alloc_for_decode_prealloc_hisparse, free_member_rows, handle_bootstrap_failure, _release_finished_req, _get_total_uncached_sizes

关键源码片段

python/sglang/srt/managers/schedule_batch.py

核心数据模型变更入口：ReqKvInfo 常驻化、is_holding_kv 单一谓词、断言收紧都在此定义

```
@dataclasses.dataclass(slots=True, kw_only=True)
class ReqKvInfo:
    # Device KV a request holds outside the prefix cache.
    # 对象在 Req 整个生命周期内常驻；释放时用 mark_released() 清零而不是置 None
    kv_allocated_len: int = 0
    swa_evicted_seqlen: int = 0

    @property
    def is_released(self) -> bool:
        # 两个字段同时归零即视为已释放，作为释放断言的一侧
        return self.kv_allocated_len == 0 and self.swa_evicted_seqlen == 0

    def mark_released(self) -> None:
        # 清零而不是删除对象，后续分配路径直接覆盖字段
        self.kv_allocated_len = 0
        self.swa_evicted_seqlen = 0


class Req(ReqDllmMixin):
    def __init__(self, ...):
        ...
        # For req-level memory management
        self.kv_committed_len = 0
        # eager 创建：三个 if req.kv is None 分配分支得以折叠
        self.kv = ReqKvInfo()
        ...

    @property
    def is_holding_kv(self) -> bool:
        # KV 所有权判定的单一事实源：req_pool_idx 是否为 None
        return self.req_pool_idx is not None
```

python/sglang/srt/mem_cache/common.py

所有请求结束时的 KV 释放主路径，生命周期不变式与 mark_released 语义在此落地

```
def release_kv_cache(req: Req, tree_cache: BasePrefixCache, is_insert: bool = True):
    # 生命周期不变量：不持有 KV（req_pool_idx 为空）当且仅当 kv 已清零
    assert (not req.is_holding_kv) == req.kv.is_released

    # MambaRadixCache 可能先分配 mamba state 再分配 KV cache
    if not req.is_holding_kv:
        assert (
            tree_cache.supports_mamba()
        ), "Only MambaRadixCache allow freeing before alloc"
```

```
        if req.mamba_pool_idx is not None:
            tree_cache.req_to_token_pool.mamba_allocator.free(
                req.mamba_pool_idx.unsqueeze(-1)
            )
            req.mamba_pool_idx = None
        return

    effective_kv_committed_len = req.effective_kv_committed_len()
    tree_cache.cache_finished_req(
        req,
        is_insert=is_insert and not getattr(req, "skip_radix_cache_insert", False),
        kv_len_to_handle=effective_kv_committed_len,
    )

    # StreamingSession.cache_finished_req 内部处理 speculative tail trim 后
    # 会设置 req_pool_idx 为 None，因此这里再次校验不变量
    assert (not req.is_holding_kv) == req.kv.is_released
    if not req.is_holding_kv:
        return

    start_p, end_p = effective_kv_committed_len, req.kv.kv_allocated_len
    _release_overallocated_kv_indices(req, start_p, end_p, tree_cache)

    # 若前缀缓存不管理 mamba 状态，则在此释放
    if isinstance(tree_cache.req_to_token_pool, HybridReqToTokenPool) and (
        not tree_cache.supports_mamba()
    ):
        assert (
            req.mamba_pool_idx is not None
        ), "mamba state is freed while the tree cache does not manage mamba states"
        tree_cache.req_to_token_pool.free_mamba_cache(req)

    tree_cache.req_to_token_pool.free(req)
    # 清零而不是置 None：ReqKvInfo 保留，下次分配直接写字段
    req.kv.mark_released()
```

评论区精华

本 PR 无任何 review 评论（comments_count=0、review_comments_count=0），作者直接合并。可以从 commit 演进还原设计意图：`eager ReqKvInfo; presence via req_pool_idx` 确立核心方案，`fix test fakes for eager kv; simplify dllm reuse check` 展示分支折叠的收益，两次 merge main 保证与主干同步。

- 暂无高价值评论线程

风险与影响

- 风险：风险：

- release_kv_cache 与 _release_finished_req 是每个请求结束的必经路径，断言收紧后，任何“设置了 req_pool_idx = None 但漏调 mark_released()”的路径会直接触发 AssertionError。这是刻意暴露存量缺陷，但也意味着仓库内其他未被本 PR 覆盖的 req.kv = None 直接赋值点（例如某些硬件后端或实验模块）可能立即暴露。
- ReqKvInfo 常驻使“释放后仍访问 kv 字段”从 AttributeError 变成读到零点 / 旧值，可能掩盖访问已释放状态的逻辑错误；不过 mark_released() 会把两字段归零，误读会得到 0。
- SessionSlot 常驻一个 ReqKvInfo，slot 数量多时存在微量内存增加，可忽略。
- 涉及 10 个源文件、横跨调度、缓存、PD 分离、beam search、PP 多条路径，虽有 run-ci 全量验证，但边缘路径（如 DSV4-NPU、Mamba、hi sparse 组合）覆盖仍可能不足。
- 影响：对用户无直接行为变化（纯内部重构）。对代码库的收益是“持有 KV”的判定从两个平行 Optional 收敛为单一谓词，后续把 req_pool_idx、mamba state、retraction_backup 等收进 ReqKvInfo 的系列重构（#37094、#37108、#37164）均建立在这一数据模型之上。对所有接触 Req/SessionSlot 的开发者：新增的 is_holding_kv 语义必须遵守，释放必须走 mark_released()。
- 风险标记：核心调度路径变更，跨模块联动重构，断言收紧暴露存量缺陷，释放路径遗漏 mark_released 风险

关联脉络

- PR #37094 [mem_cache] Move req_pool_idx into ReqKvInfo: 本 PR 确立 is_holding_kv 单一谓词后，37094 将 req_pool_idx 收进 ReqKvInfo，是同一所有权收拢系列的下一步，直接依赖本 PR 的数据模型。
- PR #37108 [mem_cache] Share one ReqKvInfo between a streaming session slot and its request: 本 PR 让 ReqKvInfo 常驻并支持清零复用，37108 才得以让 slot 与 req 共享同一个 ReqKvInfo。
- PR #37164 [mem_cache] Move mamba state and retraction_backup into ReqKvInfo: 继续把 mamba state、retraction_backup 收进 ReqKvInfo，涉及本 PR 修改的 schedule_batch.py、streaming_session.py 等文件。