PR #37167 完整报告

sgl-project/sglang

[mem_cache] Make release, row-reuse asserts, and presence checks read the KV record

合并时间：2026-09-01 03:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37167

执行摘要

- 一句话：KV 释放、复用断言与存在性检查统一走 KV record
- 推荐动作：值得精读。重点看 memory_pool.py 的 alloc / free 断言设计与 scheduler_pp_mixin.py 的统一释放路径。本 PR 是“断言应基于记录状态而非派生字段”的样板：holds_kv 与 kv_allocated_len 的语义可以直接复用到后续 unified-memory 改造中。若团队正在做 KV ownership 或 radix cache 相关重构，建议合入后密切观察 dllm / disagg / pp 路径是否有新断言触发。

功能与动机

关联 Issue #37094 先把 req_pool_idx 从 Req / SessionSlot 移入 ReqKvInfo，让记录自己回答存在性（is_held 与 is_released 配对）。本 PR body 指出三类“绕过记录契约”的存量写法：PP profiling 循环手写释放（raw slot slice free + pool mamba free + row free），“跳过 kv_len_to_handle 和 per-cache finished paths”；行复用断言从 scheduler 字段（inflight_middle_chunks、kv_committed_len）推断，导致“已释放的行能以 stale state 通过断言”；13 处存在性检查直接拼写 req_pool_idx is not None。目标是让 release、row-reuse 断言与 presence 检查统一读 KV record，停止绕过契约。

实现拆解

1. 统一 PP profiling 释放路径：python/sglang/srt/managers/scheduler_pp_mixin.py 的 profile_and_init_predictor 中，原来手写“取 kv_indices → token_to_kv_pool_allocator.free → free_mamba_cache → pool.free → mark_kv_released”五步释放，改为一行 release_kv_cache(req, self.tree_cache, is_insert=False)。等价性由作者在 PR body 中论证：这些 profiling 请求满足 cache_protected_len == 0、kv_committed_len == kv_allocated_len == extend_range.end、last_node is None，因此 radix / chunk / mamba 三种 cache 的释放范围一致，且 mamba 所有权改由 cache contract 决定而不是总走 pool。
2. 收紧行复用断言：python/sglang/srt/mem_cache/memory_pool.py 的 ReqToTokenPool.alloc / free 与 python/sglang/srt/disaggregation/decode.py 的 DecodeReqToTokenPool.alloc / free 中：复用行判定从 req_pool_idx is not None 改为 holds_kv；断言从 inflight_middle_chunks > 0 or kv_committed_len > 0 改为 kv_allocated_len > 0（复用行必须实际携带已分配 KV）。同时删除长期放宽的“only one chunked request”断言及其注释残留——单 batch 单 chunked 请求自 #20476 起由

batch composition 强制，此处不再重复校验。

3. presence 检查统一到 holds_kv：共 13 处 req_pool_idx is not None / is None 改为 holds_kv / not holds_kv，涉及 sparse_coordinator.py（on_request_begin / on_request_end）、dsv4_req_to_token_pool.py（set_c128_prefix_pages / alloc 的 fresh 判定）、allocation.py（alloc_for_extend 的 DLLM reuse_kv）、scheduler.py（process_pending_chunked_abort）、pool_stats_observer.py（active_pool_idxs）、radix_cache_cpp.py（cache_finished_req / cache_unfinished_req 的 assert）等。哨兵域检查（decode offload manager 中 req_pool_idx is None or == -1）保留 raw field read，不改为 holds_kv。
4. 修复 benchmark 与测试夹具：python/sglang/benchmark/one_batch.py 的 correctness-test 路径从 req.req_pool_idx 改为 req.kv.req_pool_idx（字段在 #37094 已迁移）；测试侧 test_decode_retraction_backup.py、test_dllm_fdfo_kv_reuse.py、test_minicpm_sparse_cache.py 把触及 holds_kv 的 SimpleNamespace fake 升级为真实 ReqKvInfo。
5. 测试与 CI 配套：本 PR 无新增针对性单测，系统性回归依赖既有用例 + CI。作者在 Issue 评论中多次 /rerun-test 了 test_pp_single_node.py、test_disaggregation_basic.py、test_dllm_batching_fdfo.py、test_openai_completion_rust.py，均通过。

关键文件：

- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 ReqToTokenPool.alloc, ReqToTokenPool.free）：核心变更：ReqToTokenPool 的 alloc / free 断言与存在性判断从原始字段转向 holds_kv + kv_allocated_len，是全 PR 语义核心。
- python/sglang/srt/managers/scheduler_pp_mixin.py（模块 调度器；类别 source；类型 dependency-wiring；符号 profile_and_init_predictor）：PP dynamic-chunk profiling 的手写三步释放改为统一走 release_kv_cache，消除绕行路径。
- python/sglang/srt/disaggregation/decode.py（模块 解码池；类别 source；类型 core-logic；符号 DecodeReqToTokenPool.alloc, DecodeReqToTokenPool.free）：DecodeReqToTokenPool 同步收紧复用断言，并删除长期放宽的 single-chunked assert 及其注释残留。
- python/sglang/srt/mem_cache/radix_cache_cpp.py（模块 缓存树；类别 source；类型 core-logic；符号 cache_finished_req, cache_unfinished_req）：cache_finished_req / cache_unfinished_req 的 assert 改用 holds_kv，保持缓存树侧与记录语义一致。
- python/sglang/srt/mem_cache/allocation.py（模块 内存分配；类别 source；类型 core-logic；符号 alloc_for_extend）：DLLM 场景的 reuse_kv 存在性判断改用 holds_kv。
- python/sglang/srt/mem_cache/sparsity/core/sparse_coordinator.py（模块 稀疏缓存；类别 source；类型 core-logic；符号 on_request_begin, on_request_end）：稀疏缓存的请求注册与清理以 holds_kv 为准，避免在无 KV 时误操作。
- python/sglang/srt/hardware_backend/npu/dsv4/dsv4_req_to_token_pool.py（模块 NPU 适配；类别 source；类型 core-logic；符号 set_c128_prefix_pages, alloc）：DSV4 C128 前缀页安装与 fresh 行判定改用 holds_kv，避免绕过记录。

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 process_pending_chunked_abort）：process_pending_chunked_abort 中判断请求是否仍持有 KV 改用 holds_kv。
- python/sglang/benchmark/one_batch.py（模块 基准工具；类别 source；类型 core-logic；符号 prepare_extend_inputs_for_correctness_test）：修复字段迁移后遗留的 stale read，正确性测试路径改读 req.kv.req_pool_idx。
- python/sglang/srt/managers/scheduler_components/pool_stats_observer.py（模块 观测器；类别 source；类型 core-logic；符号 active_pool_idxs）：活跃行统计从 raw 字段改为 holds_kv，保证统计口径与 KV record 一致。
- test/registered/unit/mem_cache/test_decode_retraction_backup.py（模块 单测；类别 test；类型 test-coverage）：触及 holds_kv 的测试 fake 从 SimpleNamespace 升级为真实 ReqKvInfo。
- test/registered/unit/mem_cache/test_dllm_fdfo_kv_reuse.py（模块 单测；类别 test；类型 test-coverage）：DLLM fdfo KV 复用测试同步到 record 语义。
- test/registered/unit/layers/test_minicpm_sparse_cache.py（模块 单测；类别 test；类型 test-coverage）：稀疏缓存测试夹具对齐 holds_kv 语义。

关键符号：ReqToTokenPool.alloc, ReqToTokenPool.free, DecodeReqToTokenPool.alloc, DecodeReqToTokenPool.free, profile_and_init_predictor, SparseCoordinator.on_request_begin, SparseCoordinator.on_request_end, DSV4ReqToTokenTablesMixin.set_c128_prefix_pages, alloc_for_extend, Scheduler.process_pending_chunked_abort, PoolStatsObserver.active_pool_idxs, RadixCacheCPP.cache_finished_req, RadixCacheCPP.cache_unfinished_req, prepare_extend_inputs_for_correctness_test

关键源码片段

python/sglang/srt/mem_cache/memory_pool.py

核心变更：ReqToTokenPool 的 alloc / free 断言与存在性判断从原始字段转向 holds_kv + kv_allocated_len，是全 PR 语义核心。

```
def alloc(self, reqs: list[Req]) -> Optional[List[int]]:
    # 复用行判定：直接读 KV record 的 holds_kv（即 req_pool_idx is not None），
    # 而不是从 scheduler 字段推断 chunked continuation。
    # holds_kv 与 req_pool_idx is not None 完全等价，
    # 但语义上把“是否持有 KV”的答案收敛到记录自身。
    reusing = [i for i, r in enumerate(reqs) if r.kv.holds_kv]

    # 关键收紧：复用行必须携带已分配的 KV。
    # 旧检查用 kv_committed_len > 0 推断，但 kv_committed_len 在 release
    # 时不会清零，已释放的行仍可能带着旧值通过断言；
    # kv_allocated_len 才是“当前持有多少 KV”的 memory-layer 事实。
    assert all(
        reqs[i].kv.kv_allocated_len > 0 for i in reusing
    ), "a reused row must carry allocated KV"
```

```
    # 只为真正的新行分配槽位，已持有行的 req_pool_idx 保持不变
    select_index = self.alloc_rows(len(reqs) - len(reusing))
    if select_index is None:
        return None
    offset = 0
    for r in reqs:
        if not r.kv.holds_kv:
            r.kv.req_pool_idx = select_index[offset]
            offset += 1
    return [r.kv.req_pool_idx for r in reqs]


def free(self, req: Req):
    # free 同样以记录为准：只有真正持有行的请求才能释放
    assert req.kv.holds_kv, "request must have req_pool_idx"
    self.free_rows([req.kv.req_pool_idx])
    req.kv.req_pool_idx = None
```

python/sglang/srt/managers/scheduler_pp_mixin.py

PP dynamic-chunk profiling 的手写三步释放改为统一走 release_kv_cache，消除绕行路径。

```
# 释放 KV 与 Mamba cache：统一走 release_kv_cache，而不是手写三步释放。
# 手写路径绕过了 kv_len_to_handle 与 per-cache finished 逻辑，
# 且 mamba 所有权总是走 pool，不符合 cache contract。
# 在这些 profiling 请求上（cache_protected_len == 0、
# kv_committed_len == kv_allocated_len == extend_range.end、last_node is None），
# 三种 cache（radix / chunk / mamba）的释放范围完全相同，
# 因此这里的等价替换是安全的。
if req.kv.holds_kv:
    release_kv_cache(req, self.tree_cache, is_insert=False)
```

评论区精华

本 PR 无 review 评论（review_comments_count 为 0），以下要点提炼自 PR body 的设计论证：

- 行复用断言应读记录而非推断：作者指出 kv_committed_len 在 release 时不会被清零，旧断言会让已释放的行带着 stale state 通过检查；改为 kv_allocated_len > 0 后，语义变为“复用行必须实际持有 KV”。
- PP profiling 释放等价性：统一到 release_kv_cache 前逐项论证了 cache_protected_len == 0、kv_committed_len == kv_allocated_len == extend_range.end、last_node is None，确保 radix / chunk / mamba 三种 cache 的释放范围一致，mamba 所有权改由 cache contract 决定。
- presence 与 sentinel 边界：13 处存在性检查统一为 holds_kv，但 decode offload manager 里 req_pool_idx is None or == -1 的哨兵域检查保留 raw read，避免两种语义混淆。

- 行复用断言应读记录而不是推断 (design): 改为直接检查 kv_allocated_len > 0，语义变为“复用行必须实际持有已分配 KV”，并同步删除 single-chunked assert 及其注释残留。
- PP profiling 释放路径统一到 release_kv_cache 的等价性 (design): 在这些 profiling 请求上（cache_protected_len == 0、kv_committed_len == kv_allocated_len == extend_range.end、last_node is None），radix / chunk / mamba 三种 cache 的释放范围相同，mamba 所有权改由 cache contract 决定，等价替换成立。
- presence 检查与 sentinel 检查的边界 (design): presence 语义交给 ReqKvInfo 的 holds_kv，哨兵域检查保留 raw field read，避免两种语义混淆。

风险与影响

- 风险:
  1. 分配断言收紧可能暴露违规路径：memory_pool.py 与 decode.py 的 alloc 现在要求复用行 kv_allocated_len > 0。由于 kv_committed_len 在 release 时不清零，旧逻辑下已释放行可携带 stale state 通过检查；若存在“释放后同一调度轮内复用同一 Req”的合法路径（如 DLLM fdfo 场景），新断言会直接触发。CI 已重跑 dllm / pp / disaggregation 相关用例，但边界场景仍需留意。
  2. PP 释放等价性依赖前置条件：scheduler_pp_mixin.py 统一改走 release_kv_cache 的正确性建立在 4 个条件上；未来若 PP profiling 引入 radix 命中或 cache 保护（last_node is not None），释放范围将不再等价，需要重新论证。
  3. 删除 single-chunked assert 依赖外部约束：该保护被移除的前提是 #20476 已在 batch composition 强制执行单 chunked 请求，若该约束后续被放松，此处的防御会缺失。
  4. presence 与 sentinel 语义边界：13 处统一改用 holds_kv 后，任何新增代码若混淆“记录存在性”与“-1 哨兵”，可能在 decode offload 路径上误判。作者有意保留哨兵域 raw read，但边界仍靠约定维持。
  5. 回归覆盖依赖 CI：三处测试均为 fixture 更新，缺乏针对新断言的定向单测，系统性风险主要靠现有集成测试兜底。 - 影响：对用户无可见功能变化。影响面集中在内部 KV 生命周期语义：调度器（scheduler.py、scheduler_pp_mixin.py、pool_stats_observer.py）、内存池（memory_pool.py）、disaggregation 解码池（decode.py）、radix 缓存树（radix_cache_cpp.py）、NPU DSV4 适配、稀疏缓存协调器以及 benchmark 工具。对团队的价值是确立 ReqKvInfo（KV record）作为分配、释放、存在性判定的唯一入口，消除多处绕过记录契约的派生读取，为后续 unified-memory 与 KV ownership 演进降低语义漂移风险；同时更严格的断言有助于在开发期尽早暴露 KV 生命周期 bug。 - 风险标记：分配断言收紧可能暴露违规路径，PP 释放等价性依赖前置条件，presence 语义迁移面广，依赖 #20476 batch 层约束，测试以 CI 覆盖为主

关联脉络

- PR #37094 [mem_cache] Move req_pool_idx into ReqKvInfo: 直接前置 PR：把 req_pool_idx 移入 ReqKvInfo 并定义 is_held / is_released 谓词，本 PR 的所有改动都建立在该记录之上。