

PR #55450 完整报告

vllm-project/vllm

[Bugfix][Core] Retire Mamba states across null gaps

合并时间: 2026-09-11 15:50

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55450>

执行摘要

- 一句话: 修复 Mamba 状态退休被 null 空洞阻断导致的状态泄漏
- 推荐动作: 值得精读。推荐重点看三处: (1) `MambaManager._remove_blocks_in_range` 的“水位线 + 长度夹紧 + 自后向前跳过 null”三元组合, 这是修复泄漏同时保证幂等与摊还成本的核心, 也是理解 align 模式“位置身份不稳定”假设 (见 `has_positionally_stable_blocks`) 的最佳样本; (2) `pop_blocks_for_free` 与 `__init__` 的成对清理, 理解请求生命周期与水位线一致性的耦合; (3) `test_mamba_retirement_bounds_prefill_states` 用参数化回放把“峰值状态数上界”写成不变量的做法, 值得作为其他长上下文内存泄漏修复的测试模板。读的时候同时留意 `last_block = min(last_block, len(blocks))` 这一静默截断, 以及 E2E 期望值变化的语义; 建议合入后补一条真实模型的长 prefill 端到端回归 (断言块池峰值与输出正确性), 以弥补当前以合成回放为主、无自动评审的验证缺口。

功能与动机

PR body 明确指出问题: "Fix align-mode Mamba retirement stopping at null gaps and leaking older states. Track the retired prefix to avoid rescanning it."。作者同时界定了与相邻工作的边界: "Split from #55435. #54076 changes chunk splitting; #53803 changes checkpoint retention. Neither fixes this leak.", 即 chunk 拆分策略与 checkpoint 保留策略的改动都不会修复这个泄漏。影响面由数据佐证: DGX Spark 上 manager 相关测试在 main (144e79c) 为 3 failed / 19 passed, 本 PR 后 22 passed; 真实 GLM-5.3-Flash TP2/MTP5 cache layout 回放中每组峰值保留块 71 → 9。

实现拆解

- 变更入口与问题定位: `MambaManager.remove_skipped_blocks()` 在 align 模式下会算出应当退休的块范围, 并委托 `SingleTypeKVCacheManager._remove_blocks_in_range()`。基类实现基于“块连续、无空洞”的假设, 遇到 Mamba prefill 在相邻状态之间留下的 null 块时无法继续推进, 空洞之后更老的状态块因此长期泄漏。本 PR 在 `MambaManager` 中新增同名 override, 只在 `mamba_cache_mode == "align"` 时接管, 其余模式第一行直通 `super()`, 兼容边界清晰。
- 新增退休水位线数据结构: `MambaManager.__init__` 的 align 分支新增 `self._num_retired_blocks: dict[str, int] = {}`, 记录每个请求已经退休到的块下标。它同时承担两个职责: (a) 避免每个调度步都从下标 0 重扫整个已退休前缀, 把 480k token 长 prefill 下的成本摊还到“新增区间”; (b) 作为幂等性保证, 使同一区间被重复请求退休时不会

二次释放仍在使用的块。

3. 核心逻辑改造：新方法分三步——先用 `first_block = max(first_block, watermark)` 夹紧下界、`last_block = min(last_block, len(blocks))` 夹紧上界，空区间直接返回；再自后向前遍历区间，`continue` 跳过 `null` 空洞、对非空块收集进 `freed` 并写回 `_null_block`；最后一次性 `block_pool.free_blocks(freed)` 并把水位线推进到 `last_block`。关键点是“跳过空洞而不是停在空洞”，以及自后向前的扫描方向——`align` 模式下 `_relocate_speculative_block` 会把投机块从中间位置摘除并 `append` 到列表尾，块下标空间与列表长度可能错位，反向扫描加长度夹紧避免越界并优先处理尾部新块。
4. 生命周期清理：`pop_blocks_for_free()` 在 `align` 分支补上 `self._num_retired_blocks.pop(request_id, None)`，与 `last_state_block_idx`、`_checkpoints`、`_producer_partial_tail_reqs`、`_pending_boundary_state_offloads` 一同清理。若不清理，请求 ID 被复用时会带着旧水位线进入新请求，导致退休被静默跳过，从“泄漏”变成“永不回收”。
5. 测试与验证配套：`tests/v1/core/test_single_type_kv_cache_manager.py` 新增 `test_mamba_retirement_crosses_null_gaps`（构造 `[old, null, committed, in_flight]` 布局，连续两次退休后断言 `old` 释放、`committed/in_flight` 存活、空闲块数为 5；并在 `manager.free("r")` 后重新分配验证水位线已清理）与 `test_mamba_retirement_bounds_prefill_states`（参数化 `block_size 3584/4608`、`in_flight_chunks 0/1/2`，模拟 480k token `prefill` 的 `async` 提交滞后，断言峰值持有块恒为 `7+in_flight_chunks` 且 `free` 后全部归还）。`tests/v1/e2e/general/test_mamba_prefix_cache.py` 随行为变化更新了一行期望值。无配置、`schema` 或部署改动。

关键文件：

- `vllm/v1/core/single_type_kv_cache_manager.py`（模块 缓存管理器；类别 `source`；类型 `core-logic`；符号 `MambaManager._remove_blocks_in_range`，`MambaManager._num_retired_blocks`，`MambaManager.pop_blocks_for_free`，`MambaManager.init`）：本次修复的全部核心逻辑所在：`MambaManager` 新增 `_remove_blocks_in_range` override 与 `_num_retired_blocks` 水位线，并在 `pop_blocks_for_free` 中清理水位线；非 `align` 模式直通 `super()`，兼容边界由第一行判空决定。
- `tests/v1/core/test_single_type_kv_cache_manager.py`（模块 单元测试；类别 `test`；类型 `test-coverage`；符号 `test_mamba_retirement_crosses_null_gaps`，`test_mamba_retirement_bounds_prefill_states`）：新增两个针对性单测：
`test_mamba_retirement_crosses_null_gaps` 用 `[old, null, committed, in_flight]` 布局锁定「跨空洞退休 + 幂等不误释放 + `free` 后水位线清理」；
`test_mamba_retirement_bounds_prefill_states` 参数化回放 480k token `prefill` 并断言峰值持有块恒为 `7 + in_flight_chunks`。
- `tests/v1/e2e/general/test_mamba_prefix_cache.py`（模块 端到端测试；类别 `test`；类型 `test-coverage`；符号 `prompt_10_block_size`）：既有 E2E 期望值随退休语义变化更新一行（提交 `bf2e6f5`），是本次行为变更在外层可观测的直接证据，也是作者在评论区专门征询「是否可以改既有 E2E 期望」的对象。

关键符号: MambaManager._remove_blocks_in_range,
MambaManager.pop_blocks_for_free, MambaManager.init,
MambaManager.remove_skipped_blocks, MambaManager._relocate_speculative_block,
test_mamba_retirement_crosses_null_gaps, test_mamba_retirement_bounds_prefill_state
s

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

本次修复的全部核心逻辑所在: MambaManager 新增 _remove_blocks_in_range override 与 _num_retired_blocks 水位线, 并在 pop_blocks_for_free 中清理水位线; 非 align 模式直通 super(), 兼容边界由第一行判空决定。

```
# vllm/v1/core/single_type_kv_cache_manager.py — MambaManager (align 模式退休逻辑)
class MambaManager(SingleTypeKVCacheManager):
    def __init__(self, kv_cache_spec: MambaSpec, block_pool: BlockPool, **kwargs) -> None:
        super().__init__(kv_cache_spec, block_pool, **kwargs)
        ...
        if self.mamba_cache_mode == "align":
            # 上一步分配的状态块索引, 用于把「当前状态」前推
            self.last_state_block_idx: dict[str, int] = {}
            # 新增: 每个请求已经「退休」(释放并置空)到的块下标水位线。
            # 作用一: 避免每步从下标 0 重扫整个已退休前缀(480k token 场景下前缀极长)。
            # 作用二: 保证同一区间被重复请求退休时不会二次释放仍在使用的块。
            self._num_retired_blocks: dict[str, int] = {}
            self._allocated_block_reqs: set[str] = set()

    def _remove_blocks_in_range(
        self, request_id: str, first_block: int, last_block: int
    ) -> None:
        # 非 align 模式的状态块具有稳定的位置语义, 沿用基类实现即可。
        if self.mamba_cache_mode != "align":
            return super()._remove_blocks_in_range(request_id, first_block, last_block)
        blocks = self.req_to_blocks.get(request_id, [])
        # 下界按水位线夹紧: 已经退休过的前缀不再重复扫描 / 释放(幂等性来源)。
        first_block = max(first_block, self._num_retired_blocks.get(request_id, 0))
        # 上界按列表长度夹紧: align 模式下投机块会被摘除并 append 到列表尾部,
        # 调用方给出的块下标可能超出当前 req_to_blocks 长度, 夹紧以避免越界。
        last_block = min(last_block, len(blocks))
        if first_block >= last_block:
            return
        freed: list[KVCacheBlock] = []
        # 自后向前扫描, 并用 continue 跳过 null 而不是提前结束:
        # Mamba prefill 会在相邻状态之间留下 null 空洞, 若在空洞处停止,
        # 空洞之后更老的状态块就会一直挂在请求上直到请求结束(即本 PR 修复的泄漏)。
        for i in range(last_block - 1, first_block - 1, -1):
            if blocks[i].is_null:
                continue
```

```

        freed.append(blocks[i])
        blocks[i] = self._null_block
    if freed:
        self.block_pool.free_blocks(freed)
    # 推进水位线：下一次调用只需处理新增区间，整体摊还到新增块数。
    self._num_retired_blocks[request_id] = last_block

def pop_blocks_for_free(self, request_id: str) -> list[KVCacheBlock]:
    if self.mamba_cache_mode == "align":
        self._allocated_block_reqs.discard(request_id)
        self.last_state_block_idx.pop(request_id, None)
        # 请求结束 / 被回收时同步清理水位线，避免请求 ID 复用
        # 沿用旧进度而静默跳过退休（那会从「泄漏」变成「永不回收」）。
        self._num_retired_blocks.pop(request_id, None)
        self._checkpoints.pop(request_id, None)
        self._producer_partial_tail_reqs.pop(request_id, None)
        ...
    return super().pop_blocks_for_free(request_id)

```

tests/v1/core/test_single_type_kv_cache_manager.py

新增两个针对性单测：test_mamba_retirement_crosses_null_gaps 用 [old, null, committed, in_flight] 布局锁定「跨空洞退休 + 幂等不误释放 + free 后水位线清理」；test_mamba_retirement_bounds_prefill_states 参数化回放 480k token prefill 并断言峰值持有块恒为 7 + in_flight_chunks。

tests/v1/core/test_single_type_kv_cache_manager.py

```

def test_mamba_retirement_crosses_null_gaps():
    # 刻意构造空洞：old 是应被退休的旧状态，下标 1 是 prefill 留下的 null 空洞，
    # committed / in_flight 是必须存活的状态块。
    spec = MambaSpec(
        block_size=4, shapes=((1, 1),), dtypes=(torch.float32,), mamba_cache_mode="align"
    )
    pool = BlockPool(num_gpu_blocks=8, enable_caching=False, hash_block_size=4)
    manager = MambaManager(
        spec, block_pool=pool, enable_caching=False, kv_cache_group_id=0, scheduler_block_size=4
    )
    old, committed, in_flight = pool.get_new_blocks(3)
    manager.req_to_blocks["r"] = [old, pool.null_block, committed, in_flight]

    # 同一区间连续退休两次：第二次必须完全无副作用，
    # 即水位线阻止了对 committed 的二次释放（这是本 PR 幂等性的核心断言）。
    for _ in range(2):
        manager.remove_skipped_blocks("r", processed_computed_tokens=12)
        assert manager.req_to_blocks["r"] == [
            pool.null_block,
            pool.null_block,
            committed,

```

```

        in_flight,
    ]
    assert old.ref_cnt == 0
    assert committed.ref_cnt == in_flight.ref_cnt == 1
    assert pool.get_num_free_blocks() == 5

# free 会走 pop_blocks_for_free 清理水位线；随后重新分配并再次退休，
# 断言首块被置空，证明旧水位线没有残留（否则这里会因夹紧而直接 return）。
manager.free("r")
manager.req_to_blocks["r"] = pool.get_new_blocks(2)
manager.remove_skipped_blocks("r", processed_computed_tokens=5)
assert manager.req_to_blocks["r"][0].is_null
assert manager.req_to_blocks["r"][1].ref_cnt == 1

@pytest.mark.parametrize("block_size", [3584, 4608])
@pytest.mark.parametrize("in_flight_chunks", [0, 1, 2])
def test_mamba_retirement_bounds_prefill_states(block_size, in_flight_chunks):
    # 复刻真实配置：3584/4608 的块、5 个投机块、8192 token 的 prefill chunk。
    spec = MambaSpec(
        block_size=block_size, shapes=((1, 1),), dtypes=(torch.float32,),
        mamba_cache_mode="align", num_speculative_blocks=5,
    )
    pool = BlockPool(num_gpu_blocks=1000, enable_caching=True, hash_block_size=256)
    manager = MambaManager(
        spec, block_pool=pool, enable_caching=True, kv_cache_group_id=0,
        scheduler_block_size=block_size,
    )
    initial_free = pool.get_num_free_blocks()
    chunk = 8192 // block_size * block_size
    peak_held = 0
    # 回放 480k token 的分块 prefill，并模拟 async 调度下提交进度落后若干 chunk。
    for target in range(chunk, 480031 + 1, chunk):
        computed = target - chunk
        committed = max(0, computed - in_flight_chunks * chunk)
        manager.remove_skipped_blocks("r", committed)
        manager.get_num_blocks_to_allocate("r", target + 5, [], computed, computed, target)
        manager.allocate_new_blocks("r", target + 5, target)
        peak_held = max(peak_held, initial_free - pool.get_num_free_blocks())

    # 5 个投机块 + 当前 / 上一步状态 + 有界的 in-flight 状态 = 7 + in_flight_chunks。
    assert peak_held == 7 + in_flight_chunks
    manager.free("r")
    assert pool.get_num_free_blocks() == initial_free

```

tests/v1/e2e/general/test_mamba_prefix_cache.py

既有 E2E 期望值随退休语义变化更新一行（提交 bf2e6f5），是本次行为变更在外层可观测的直接证据，也是作者在评论区专门征询「是否可以改既有 E2E 期望」的对象。

```
# tests/v1/e2e/general/test_mamba_prefix_cache.py
"prompt_10_block_size": TestConfig(
    num_prompt_tokens=560 * 10,
    num_generated_tokens=10,
    num_accepted_tokens=4,
    step_actions=[
        ...,
        StepAction(
            560 * 10,
            4,
            # if a 分支第 5 个元素由 1 变 0: 该步的第 5 号 Mamba 状态块现在被正确退休
            # （置为 null 并归还块池），而不再残留到请求结束——这正是修复生效的观测点。
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1],
            if a
            else [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1],
            (9, 10),
            (-1, -1),
        ),
    ],
),
```

评论区精华

本 PR 22 条评论中绝大多数是 CI 触发 / 重试与 rebase 提醒，真正的技术交锋有三处。

- 1) 作者主动求审与区域耦合：lucamotz 直接点名刚刚合入同区域代码的维护者——"@ZJY0516 quick ping because you just merged the same area of code into main, and I just rebased onto that. You are probably very familiar with this right now."，说明 Mamba 退休路径正处在高频改动区域，ZJY0516 随后给出 APPROVED。
- 2) 能否修改既有 E2E 期望值：lucamotz 询问 "would it be okay for me to update the E2E test expectations as part of this PR? I believe the failures are caused by the intentional change in Mamba block retirement, but I don't want to overstep if updating those tests is something you'd prefer a maintainer to handle."，随后自查先例并自行更新（提交 bf2e6f5 "[Test] Update Mamba async retirement block expectation"）。这是本 PR 唯一被明确讨论的“行为变更是否可接受”问题，结论是接受，因为期望值变化本身就是修复生效的证据。
- 3) 第三方独立复现：BruceFeIix 在 v0.29.0、RTX 5090/WSL2、Qwen3.5 架构 27B ModelOpt NVFP4 权重、无 MTP/ 投机解码的条件下复现了同一问题，并给出对比：同一 237000 token 冷请求，调度 prefill token 437704 → 237000、prefill 步数 141 → 77、computed-token 进度回滚 2 → 0、TTFT 148.952s → 85.932s；还提到“Disabling async scheduling on unmodified v0.29.0 also avoided the long-prefill slowdown”，指向问题与 async 提交进度和退休交互有关。

另外 claude[bot] 因 PR 来自 fork 而禁用自动评审，最终只有一位人类 reviewer 批准。

- 跨 null 空洞退休的幂等性与水位线语义 (correctness): 实现被采纳并由 ZJY0516 批准；水位线在 pop_blocks_for_free 中清理，测试尾段验证了 free 后水位线不残留。

- 是否允许在本 PR 中修改既有 E2E 测试期望值 (testing): 接受在修复型 PR 内同步更新受影响的 E2E 期望值; 维护者未提出异议, 最终给出 APPROVED。
- 与 #54076 / #53803 / #55435 的职责边界 (design): 变更范围被刻意收敛为单函数 override + 水位线, 降低与并行 Mamba 改造的冲突面 (这也解释了作者需要多次 rebase main)。
- 第三方独立复现: 无投机解码场景同样受益 (other): 印证收益不只来自 MTP/ 投机路径, 而是 async 提交进度与退休扫描交互的系统性问题; 作者与 reviewer 均未反驳。

风险与影响

- 风险:
 1. 过度释放的正确性风险 (最高): `_remove_blocks_in_range` 直接触碰块池释放路径, 一旦把仍在被引用的状态块置空并归还, 后续会被重新分配, 表现为静默错误输出而非崩溃。缓解证据: `test_mamba_retirement_crosses_null_gaps` 对同一区间连续调用两次并断言 `committed.ref_cnt == in_flight.ref_cnt == 1`、`pool.get_num_free_blocks() == 5`, 锁定幂等性; `test_mamba_retirement_bounds_prefill_states` 断言 `peak_held == 7 + in_flight_chunks` 且 `free` 后空闲块完全归还。
 1. 水位线残留导致的“反向泄漏”: 若某条路径重置 `req_to_blocks[request_id]` 却未经过 `pop_blocks_for_free`, `_num_retired_blocks` 会保留旧值, `first_block` 被夹紧到水位线之上后直接 `return`, 退休被静默跳过。测试尾段 (`manager.free("r")` 后重新分配 2 块并断言 `req_to_blocks["r"][0].is_null`) 正是覆盖这一路径, 但仅覆盖单一入口。
 2. 下标空间与列表长度错位的静默截断: `last_block = min(last_block, len(blocks))` 会把超出当前列表长度的退休请求静默吞掉。align 模式下 `_relocate_speculative_block` 会把投机块 `append` 到列表尾部并在中间位置留 `null`, 说明块下标空间与 `req_to_blocks` 长度确实可能不一致; 若未来调用方依赖被截断的范围, 会重新引入有界泄漏 (不致命, 但需要注释或断言约束)。
 3. 行为变更对既有测试与下游期望的冲击: `tests/v1/e2e/general/test_mamba_prefix_cache.py` 中 `prompt_10_block_size` 场景第 5 个状态块由保留变为退休 (1 → 0), 说明 align 模式的留存策略有可观测变化, 任何对块占用数做硬编码期望的下游测试或容量规划都需要同步检查。
 4. 测试覆盖偏“结构性”而非“端到端模型正确性”: 新增单测是纯 manager 级的合成回放 (固定 async 滞后模型 `in_flight_chunks * chunk`), E2E 只改了一行期望值, 没有新增一条“长 prefill + 真实模型 + 输出正确性”的端到端断言。作者给出的显存收益 (2006.44 MiB → 254.34 MiB/group) 来自本机回放, 且 PR 来自 fork、无自动评审、AI 辅助生成 (OpenAI Codex co-author), 审阅冗余度偏低。
 5. 性能方向总体为正: 水位线避免了每步重扫已退休前缀, 属于纯收益; 未观察到新增热路径开销。- 影响: 用户侧: 使用 align 模式 Mamba / 混合架构模型 (如 GLM-5.3-Flash) 并开启 chunked prefill、async scheduling、prefix caching 的长上下文用户直接受益——作者回放中每组 Mamba 状态池占用从 2006.44 MiB 降到 254.34 MiB (每组回收约 1,752 MiB), 第三方在无投机解码条件下测得长 prefill TTFT 从 148.952s 降到 85.932s、调度 token 量减半, 意味着同等显存下可承载更长上下文或更大并发。

系统侧：改动集中在 `vllm/v1/core/single_type_kv_cache_manager.py` 的 `MambaManager`，非 align 模式直通基类，对纯注意力模型、普通 Mamba 模式、KV connector 的接口均无影响；但块释放时序变化会轻微改变块池复用顺序，理论上会影响 prefix cache 的命中 / 淘汰细节（作者提交 `bf2e6f5` 更新 E2E 期望值即是这一影响的直接体现）。

团队侧：这是 `kv-cache-manager` 区域在 #537xx/#54xxx 系列 Mamba align 模式改造之后的又一次关键修补，与 #54736（SimpleCPU 细粒度 hybrid 前缀命中）、#56214（DeepSeek-V4.1-Flash 的 state/KV 池改造）共享同一套状态池机制；后续任何改动 `remove_skipped_blocks` / `pop_blocks_for_free` / 投机块搬迁的 PR 都必须重新审视水位线语义。

- 风险标记：核心路径变更，块释放正确性风险，水位线残留风险，E2E 期望值行为变更，缺少真实模型端到端回归，AI 辅助且仅单次人工审批

关联脉络

- PR #54736 [Feature][SimpleCPU] Load fine-grained hybrid prefix hits: 同样修改 `vllm/v1/core/single_type_kv_cache_manager.py`（并波及 `simple_kv_offload_manager` 与 `kv_cache_coordinator`），属于同一套 hybrid / 外部前缀命中与块留存的改动线；本 PR 的退休语义变化会直接影响这类前缀命中的块复用期望。
- PR #49675 [Bugfix][Core] Stop zero-progress preemption cascades for deferred KV frees: 同属 v1 核心块释放生命周期：那一次修复的是延迟释放栅栏未推进导致的零进展抢占级联，本 PR 修复的是退休扫描在 null 空洞处停住导致的释放不足，二者都在约束「块何时真正归还块池」。
- PR #56214 [Model] Support DeepSeek-V4.1-Flash: 改动涉及 `vllm/v1/core/kv_cache_utils.py` 与 `vllm/v1/simple_kv_offload/manager.py`，同属 Mamba/ 混合架构状态池与 KV 布局机制；本 PR 在 body 中也以 GLM-5.3-Flash TP2/MTP5 的真实 cache layout 做回放，说明该区域正被多模型支持工作密集改造。