PR #25770 完整报告

sgl-project/sglang

[Bug][Mamba] Fix LRU list reference cycle leak in mamba_radix_cache

合并时间：2026-05-21 01:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25770

执行摘要

- 一句话：修复 Mamba LRU 链表节点引用循环内存泄漏
- 推荐动作：建议合并。该 PR 定位精准、修复简洁、验证充分，属于典型的高收益低风险修复。值得学习的是其系统性的诊断方法（malloc_trim、gc.collect、DEBUG_SAVEALL）和 reviewer 提醒的统一修复范围。

功能与动机

修复 scheduler RSS 在混合 Mamba 模型（如 Qwen3-Next）持续负载下线性增长的内存泄漏，实测约为 3.7 GB/h。作者通过 malloc_trim、flush_cache、gc.collect 等诊断手段定位到 gen-2 GC 可回收约 1.9 GB，确认是引用循环导致。

实现拆解

1. 在 python/sglang/srt/mem_cache/mamba_radix_cache.py 的 LRUList._remove_node 中，在原有的邻居指针更新后，增加 setattr(node, self.prv, None) 和 setattr(node, self.nxt, None)，清除被移除节点的前后向引用。
2. 在 python/sglang/srt/mem_cache/swa_radix_cache.py 的 SWALRUList._remove_node 中，应用完全相同的修复。
3. 在 python/sglang/srt/mem_cache/unified_radix_cache.py 的 UnifiedLRUList._remove_node 中，应用类似修复，但使用直接属性赋值 node.lru_prev[pt] = None; node.lru_next[pt] = None。
4. 所有修复仅增加 2 行代码，无其他逻辑变更，不改动外部接口或行为。

关键文件：

- python/sglang/srt/mem_cache/mamba_radix_cache.py（模块 内存缓存；类别 source；类型 core-logic；符号 _remove_node）：原始 bug 所在文件，LRUList._remove_node 方法中清除 node.prev/node.next 指针以打破引用循环。
- python/sglang/srt/mem_cache/swa_radix_cache.py（模块 内存缓存；类别 source；类型 core-logic；符号 _remove_node）：SWALRUList._remove_node 存在完全相同的 bug，同步修复。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 内存缓存；类别 source；类型 core-logic；符号 _remove_node）：UnifiedLRUList._remove_node 存在完全相同的 bug，同步修复。

关键符号：_remove_node

关键源码片段

python/sglang/srt/mem_cache/mamba_radix_cache.py

原始 bug 所在文件，LRUList._remove_node 方法中清除 node.prev/node.next 指针以打破引用循环。

```
def _remove_node(self, node):
    """Helper to remove node from linked list"""
    # 更新邻居指针跳过被删除节点
    setattr(getattr(node, self.prv), self.nxt, getattr(node, self.nxt)) # node.prev.next = node.next
    setattr(getattr(node, self.nxt), self.prv, getattr(node, self.prv)) # node.next.prev = node.prev
    # 清除自指针，打断被逐出节点间的引用循环（关键修复）
    setattr(node, self.prv, None)
    setattr(node, self.nxt, None)
```

python/sglang/srt/mem_cache/swa_radix_cache.py

SWALRUList._remove_node 存在完全相同的 bug，同步修复。

```
def _remove_node(self, node):
    """Helper to remove node from linked list"""
    setattr(getattr(node, self.prv), self.nxt, getattr(node, self.nxt)) # node.prev.next = node.next
    setattr(getattr(node, self.nxt), self.prv, getattr(node, self.prv)) # node.next.prev = node.prev
    # 清除自指针，打破引用循环
    setattr(node, self.prv, None)
    setattr(node, self.nxt, None)
```

python/sglang/srt/mem_cache/unified_radix_cache.py

UnifiedLRUList._remove_node 存在完全相同的 bug，同步修复。

```
def _remove_node(self, node: UnifiedTreeNode):
    pt = self._pt
    # 从链表中摘除当前节点
    node.lru_prev[pt].lru_next[pt] = node.lru_next[pt]
    node.lru_next[pt].lru_prev[pt] = node.lru_prev[pt]
    # 清除自指针，打断引用循环
    node.lru_prev[pt] = None
    node.lru_next[pt] = None
```

评论区精华

审核者 hzh0425 在 Issue 评论中指出 swa_radix_cache 和 UnifiedRadixCache 存在相同的 _remove_node bug 模式，建议一并修复。作者 zjd0112 在下一个 commit 中同步修复了这两个文件，获得审核者批准。

- 是否应将修复扩展到 swa 和 unified radix cache (correctness): 作者同意并在后续 commit 中同步修复了这两个文件，确保所有 LRU 链表实现一致。

风险与影响

- 风险：风险极低：仅 3 个文件各增加 2 行清零操作，不改变控制流或外部行为。清零后节点若被意外复用（dangling pointer）可能导致断言失败，但现有代码中节点被移除后不会再访问其 LRU 指针（逻辑由调用方保证），因此回归风险很小。
- 影响：直接影响：修复在混合 Mamba 模型持续推理时 scheduler 的内存泄漏，长期运行可避免 OOM 或 RSS 爆炸。间接影响：所有使用 LRU 链表缓存的模块（mamba、swa、unified radix tree）都将受益于更及时的内存回收。对单次请求无行为变化。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- 暂无明显关联 PR