PR #22862 完整报告

sgl-project/sglang

Streaming session: fix retract tail leak via _free_tail

合并时间：2026-04-15 16:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22862

执行摘要

- 一句话：修复流式会话 KV 内存泄漏，统一尾部释放逻辑并移除重复处理。
- 推荐动作：该 PR 值得精读，特别是 _free_tail 的设计决策如何统一处理多种泄漏场景，以及页面对齐的重要性。关注 match_prefix 中前缀长度计算和断言，理解流式会话的只追加属性如何被强制执行。

功能与动机

流式会话在恢复时，alloc_for_extend 会覆盖 req_to_token 中旧的 KV 池索引，导致内存泄漏。旧设计在 common.py 中通过特殊分支处理推测解码尾部，但漏掉了 logit-reserve 偏移和回退重试产生的泄漏（每个回合漏 1 个 token）。PR body 指出这违反了装饰器模式，且维护复杂，因此需要统一修复。

实现拆解

1. 入口调整：在 session_aware_cache.py 的 match_prefix 方法中，修复 prefix_len 计算（移除多余的 -1 偏移），并添加断言确保流式会话的只追加属性。
2. 核心逻辑整合：新增 _free_tail 方法，在 match_prefix 中调用，用于释放 [prefix_len, kv_allocated_len) 范围内的孤儿 KV 索引。此方法自动处理页面对齐（当 page_size > 1 时），防止分页分配器损坏，并更新 slot 和 req 的长度字段。
3. 依赖清理：在 common.py 的 release_kv_cache 函数中，移除所有流式会话的特殊处理分支和导入，现在流式会话的尾部释放由 SessionAwareCache 内部处理，简化通用路径。
4. 提交演进：通过 5 个 commits 逐步完善，包括添加页面对齐逻辑、简化注释和文档，确保最终实现正确且清晰。

关键文件：

- python/sglang/srt/mem_cache/session_aware_cache.py（模块 内存缓存；类别 source；类型 core-logic；符号 match_prefix, _free_tail）：主变更文件，实现了 _freeze_tail 方法并将其集成到 match_prefix 中，修复内存泄漏的核心逻辑。
- python/sglang/srt/mem_cache/common.py（模块 通用工具；类别 source；类型 dependency-wiring；符号 release_kv_cache）：移除流式会话特殊处理，简化 release_kv_cache 函数，依赖 session_aware_cache 内部处理尾部释放。

关键符号：_free_tail, match_prefix, release_kv_cache

关键源码片段

python/sglang/srt/mem_cache/session_aware_cache.py

主变更文件，实现了 _freeze_tail 方法并将其集成到 match_prefix 中，修复内存泄漏的核心逻辑。

```
def _free_tail(self, slot: SessionSlot, req: Req, prefix_len: int):
    """Free KV in [prefix_len, kv_allocated_len) before the next
    alloc_for_extend overwrites it. The gap appears when spec
    decoding pushes allocated above committed, or when retract
    retry's logit-reserve pulls prefix_len below committed.
    Free start is ceil-aligned to page_size: PagedTokenToKVPoolAllocator
    frees by whole pages, so partial-page free would corrupt pages
    still holding committed tokens; the gap stays attached until
    release_session.
    """
    if prefix_len >= slot.kv_allocated_len:
        return # 无尾部可释放，提前返回避免不必要操作
    free_start = prefix_len
    if self.page_size > 1:
        # 对齐到页面边界，防止分页分配器损坏
        free_start = ceil_align(free_start, self.page_size)
    if free_start < slot.kv_allocated_len:
        # 获取并释放孤儿索引范围
        tail_indices = self.req_to_token_pool.req_to_token[
            slot.req_pool_idx, free_start : slot.kv_allocated_len
        ]
        self.token_to_kv_pool_allocator.free(tail_indices)
    # 更新slot和req的长度字段，反映已释放的尾部
    slot.kv_allocated_len = prefix_len
    slot.kv_committed_len = min(slot.kv_committed_len, prefix_len)
    slot.swa_evicted_seqlen = min(slot.swa_evicted_seqlen, prefix_len)
    req.kv_allocated_len = prefix_len
    req.kv_committed_len = min(req.kv_committed_len, prefix_len)
    req.swa_evicted_seqlen = min(req.swa_evicted_seqlen, prefix_len)
```

评论区精华

- 页面对齐风险：reviewer gemini-code-assist[bot] 指出 _free_tail 必须对 free_start 进行 ceil 对齐到页面边界，否则在分页分配器中可能导致内存损坏。这已在提交 a1b4dda 中通过添加 ceil_align 逻辑解决。
- 文档完整性：reviewer 建议恢复 release_session 方法中的详细 docstring，以保留关于 radix 树分裂和避免重新匹配的技术上下文，防止未来回归。但最终代码中 docstring 被简化，部分上下文可能丢失。
  - 页面对齐在 _freeze_tail 中的必要性 (correctness): 在提交 a1b4dda 中添加了 ceil_align 逻辑，确保页面边界对齐，已解决风险。

- release_session 方法文档的完整性 (documentation): 最终代码中 docstring 被简化，部分技术细节可能丢失，但核心逻辑不变，状态为部分解决。

风险与影响

- 风险：- 回归风险：核心路径 match_prefix 的变更影响所有流式会话的 KV 匹配和释放，若 _free_tail 的页面对齐逻辑错误，可能导致 KV 池损坏或内存泄漏。
- 性能风险：_free_tail 在每次 match_prefix 调用中执行，但仅在 prefix_len < kv_allocated_len 时操作，常见情况下无额外开销。
- 兼容性风险：移除 common.py 中的流式会话特殊处理，依赖该逻辑的其他模块（如非流式会话）应不受影响，但需确保 SessionAwareCache 正确接管所有流式会话尾部释放。
- 影响：- 用户影响：修复内存泄漏，提升流式会话的稳定性和资源利用率，长期运行会话不再积累孤儿 KV 索引。
- 系统影响：核心内存缓存模块（session_aware_cache）逻辑更统一，减少维护复杂度；common.py 更简洁，降低认知负担。
- 团队影响：为未来流式会话功能扩展（如新推测模式）提供更健壮的基础，但开发者需注意新的断言和 _free_tail 调用点。
- 风险标记：核心路径变更，页面对齐风险，缺少详细文档

关联脉络

- PR #22790 Refactor streaming session abort handling: 同属流式会话内存管理改进，本 PR 修复的泄漏问题与中止处理相关，涉及相同模块和设计模式。
- PR #22753 Fix streaming session busy-check double-counting via active_pool_idxs: 都修复流式会话的内存统计问题，共享 session_aware_cache 和 common.py 的变更上下文。