PR #22900 完整报告

sgl-project/sglang

trim_overshoot: cap swa_evicted_seqlen + unit test

合并时间：2026-04-16 06:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22900

执行摘要

- 一句话：修复流式会话中 Spec V2 解码超限时 SWA（滑动窗口注意力）内存泄漏问题。
- 推荐动作：该 PR 值得精读，尤其是关注 _trim_overshoot 和 _free_tail 方法如何协同维护 KV 和 SWA 状态的一致性。设计决策体现了对称性修复的重要性，对于处理流式会话中的内存管理有借鉴意义。

功能与动机

根据 PR body 描述，问题源于 _trim_overshoot（在 PR #22897 中添加）在裁剪 kv_committed_len 和 kv_allocated_len 时，未对 req.swa_evicted_seqlen 进行封顶，而 _free_tail（match_prefix 路径上的对应函数）则对所有三个字段都进行了封顶。这种不对称性导致当一轮解码超限且 SWA 正在主动驱逐时，SWA 内存池会发生泄漏。具体来说，swa_evicted_seqlen 是一个游标，如果超限将其推过裁剪边界（例如 swa_evicted = 42 但 target = 38），后续的 save_from_req 和 restore_to_req 会传播这个过时的值，导致新写入的 SWA 槽位被跳过并永久累积，形成内存泄漏。

实现拆解

1. 核心逻辑修复：在 python/sglang/srt/mem_cache/session_aware_cache.py 的 _trim_overshoot 方法中，添加一行 req.swa_evicted_seqlen = min(req.swa_evicted_seqlen, target)，确保 SWA 驱逐游标也被限制在目标边界内，与 _free_tail 方法保持一致。
2. 单元测试配套：在 test/registered/unit/mem_cache/test_streaming_session_unit.py 中新增 test_trim_overshoot_postcondition 测试函数，模拟超限场景（origin=26, finished_len=12, target=38），验证修复后 kv_committed_len、kv_allocated_len、swa_evicted_seqlen 均被正确封顶至 target，output_ids 被截断，且尾部 KV 槽位被释放。
3. 提交历史演进：提交历史显示修复过程有反复（添加、回滚、再添加），最终与主分支合并，表明可能涉及并发修改或冲突解决，但最终稳定集成。

关键文件：

- python/sglang/srt/mem_cache/session_aware_cache.py（模块 会话缓存；类别 source；类型 core-logic；符号 _trim_overshoot）：这是修复内存泄漏的核心源码文件，修改了 _trim_overshoot 方法以封顶 swa_evicted_seqlen。

- test/registered/unit/mem_cache/test_streaming_session_unit.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_trim_overshoot_postcondition）：新增单元测试，验证修复后 _trim_overshoot 方法的完整后置条件，包括 SWA 字段的封顶。

关键符号：_trim_overshoot, test_trim_overshoot_postcondition

关键源码片段

python/sglang/srt/mem_cache/session_aware_cache.py

这是修复内存泄漏的核心源码文件，修改了 _trim_overshoot 方法以封顶 swa_evicted_seqlen。

```
def _trim_overshoot(self, req: Req, finished_len: int):
    """Trim slot KV to finished_len boundary. Spec v2 may overshoot
    max_new_tokens (verify round commits M+1 at a time); next turn's
    input is output_ids[:finished_len], so positions past that must
    be released to avoid token/KV mismatch.
    """
    target = len(req.origin_input_ids) + finished_len
    self._free_kv_aligned(req.req_pool_idx, target, req.kv_allocated_len)
    req.kv_allocated_len = min(req.kv_allocated_len, target) # 封顶已分配长度
    req.kv_committed_len = min(req.kv_committed_len, target) # 封顶已提交长度
    req.swa_evicted_seqlen = min(req.swa_evicted_seqlen, target) #
    新增：封顶SWA驱逐游标，防止内存泄漏
    req.output_ids = req.output_ids[:finished_len] # 截断输出ID
```

test/registered/unit/mem_cache/test_streaming_session_unit.py

新增单元测试，验证修复后 _trim_overshoot 方法的完整后置条件，包括 SWA 字段的封顶。

```
def test_trim_overshoot_postcondition():
    """`_trim_overshoot` 后置条件：每个请求的KV字段都被封顶在 target = origin+finished_len，
    输出ID被截断，尾部KV槽位被释放。覆盖非SWA字段（kv_committed_len, kv_allocated_len,
    output_ids）
    和SWA簿记（swa_evicted_seqlen）——与 `_free_tail` 在match_
    prefix路径上强制执行的相同不变量。
    """
    page_size = 1
    req_to_token = torch.arange(128, dtype=torch.int32).reshape(1, 128)
    req_to_token_pool = SimpleNamespace(req_to_token=req_to_token, free_slots=[])
    allocator = _FakeAllocator()
    tree_cache = SessionAwareCache(_FakeInnerCache(req_to_token_pool, allocator, page_size))

    # 超限场景：origin=26, finished_len=12 -> target=38
    # committed=40（超限2），allocated=44, swa_evicted=42（> target），
    # output_ids 被超限轮扩展到14
    req = _FakeReq("session-a", req_pool_idx=0, committed=40, allocated=44)
    req.origin_input_ids = list(range(26))
    req.output_ids = list(range(14))
    req.swa_evicted_seqlen = 42 # 模拟泄漏场景：游标超过目标边界

    tree_cache._trim_overshoot(req, finished_len=12)
```

```
target = 38
assert req.kv_committed_len == target # 验证已提交长度被封顶
assert req.kv_allocated_len == target # 验证已分配长度被封顶
assert req.swa_evicted_seqlen == target # 关键断言：SWA游标也被封顶，防止泄漏
assert len(req.output_ids) == 12 # 验证输出ID被截断
# 尾部 [38, 44) 被 _free_kv_aligned 释放
assert len(allocator.freed) == 1
assert allocator.freed[0].tolist() == list(range(38, 44))
```

评论区精华

由于 review 评论为空，无公开讨论记录。但从 PR body 和提交历史看，作者 hnyls2002 自行识别问题、实施修复并添加测试，通过 CI 验证后合并。

- 暂无高价值评论线程

风险与影响

- 风险：1. 回归风险低：变更仅添加一行代码，逻辑简单直接，且与现有 _free_tail 方法对称，降低了引入新错误的风险。 2. 性能影响可忽略：min 操作开销极小，不影响核心路径性能。 3. 兼容性无影响：这是对 PR #22897 引入功能的修复，不改变外部接口或行为，仅内部状态一致性修复。 4. 测试覆盖充分：新增单元测试直接针对泄漏场景，验证了修复的有效性，但需确保测试在 CI 中稳定运行。
- 影响：1. 对用户：修复了 SWA 内存泄漏问题，提升流式会话的稳定性和资源利用率，用户无感知但系统更健壮。 2. 对系统：确保 SWA 内存池在超限场景下正确释放，避免长期运行中的内存累积，影响系统可扩展性。 3. 对团队：强化了 _trim_overshoot 与 _free_tail 之间的一致性设计模式，为后续类似修复提供参考。
- 风险标记：内存泄漏修复，状态一致性

关联脉络

- PR #22897 streaming session: trim spec v2 overshoot in cache_finished_req: 本 PR 修复了 #22897 引入的 _trim_overshoot 函数中未封顶 swa_evicted_seqlen 的问题，两者直接关联。
- PR #22862 Streaming session: fix retract tail leak via _free_tail: 涉及相同的 session_aware_cache.py 文件和内存泄漏修复，本 PR 确保 _trim_overshoot 与 _free_tail 保持对称。
- PR #22790 Refactor streaming session abort handling: 同属流式会话和内存管理模块的 PR，关注会话状态一致性和内存泄漏问题。