PR #27402 完整报告

sgl-project/sglang

[sgl] proactively release out-of-window SWA slots after chunked prefill

合并时间：2026-06-12 14:17

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27402

执行摘要

- 一句话：主动释放 SWA 窗口外 KV 槽位，优化长上下文内存使用
- 推荐动作：值得精读，尤其是组件级 hook 设计和页面边界保护逻辑（page_size 减一策略），以及如何通过环境变量平滑引入行为变更。

功能与动机

在长上下文运行中，未完成的 prefill 请求中的 SWA KV 槽位即使已滑出窗口仍被持续占用，导致 SWA 内存使用不受控。PR body 明确提出“bounding SWA pool usage for long-context runs”。

实现拆解

1. 添加基类 hook：在 TreeComponent（tree_component.py）中新增 free_out_of_window_slots 默认方法（pass），供子类按需重写。
2. SWA 组件实现：SWAComponent（swa_component.py）实现该方法，调用通用函数 free_swa_out_of_window_slots，计算应释放的槽位范围（pre_len - sliding_window_size - page_size）并调用 token_to_kv_pool_allocator.free_swa。
3. 驱动点注入：在 UnifiedRadixCache.cache_unfinished_req（unified_radix_cache.py）中，当环境变量启用时，遍历所有组件并调用 free_out_of_window_slots，结果记录到 insert_params.swa_evicted_seqlen。
4. 调度层重构：将 ScheduleBatch._evict_swa（schedule_batch.py）中的内联释放逻辑替换为调用 free_swa_out_of_window_slots，消除代码重复。
5. 测试覆盖：在 test_unified_radix_cache_unittest.py 中添加 test_swa_eager_eviction_on_unfinished_req 和 test_swa_eager_eviction_noop_when_within_window，验证释放行为和窗口内无操作。
6. 配置：在 environ.py 中新增环境变量 SGLANG_OPT_UNIFIED_CACHE_FREE_OUT_OF_WINDOW_SLOTS，默认关闭。

关键文件：

- python/sglang/srt/mem_cache/common.py（模块 缓存层；类别 source；类型 core-logic；符号 free_swa_out_of_window_slots）：新增核心函数 free_swa_out_of_window_slots，包含页面边界保护和槽位释放逻辑，是整个功能的计算核心。

- python/sglang/srt/mem_cache/unified_cache_components/swa_component.py（模块 缓存层；类别 source；类型 core-logic；符号 free_out_of_window_slots）：实现 free_out_of_window_slots hook，将 SWA 组件特定的参数传递给通用函数，并更新 insert_params。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring）：在 cache_unfinished_req 中根据环境变量驱动组件循环，是 hook 的调用入口。
- python/sglang/srt/managers/schedule_batch.py（模块 调度器；类别 source；类型 core-logic；符号 _evict_swa）：将 _evict_swa 内联逻辑重构为调用通用函数 free_swa_out_of_window_slots，消除重复代码。
- python/sglang/srt/mem_cache/unified_cache_components/tree_component.py（模块 缓存层；类别 source；类型 core-logic；符号 free_out_of_window_slots）：基类 TreeComponent 添加 free_out_of_window_slots 空方法，定义了 hook 接口。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 测试；类别 test；类型 test-coverage；符号 test_swa_eager_eviction_on_unfinished_req, test_swa_eager_eviction_noop_when_within_window）：新增两个测试用例覆盖主动释放和窗口内无操作，验证逻辑正确性。

关键符号：free_swa_out_of_window_slots, SWAComponent.free_out_of_window_slots, UnifiedRadixCache.cache_unfinished_req, ScheduleBatch._evict_swa

关键源码片段

python/sglang/srt/mem_cache/common.py

新增核心函数 free_swa_out_of_window_slots，包含页面边界保护和槽位释放逻辑，是整个功能的计算核心。

```
# python/sglang/srt/mem_cache/common.py

def free_swa_out_of_window_slots(
    req: Req,
    pre_len: int,
    *,
    sliding_window_size: int,
    page_size: int,
    req_to_token_pool: ReqToTokenPool,
    token_to_kv_pool_allocator: BaseTokenToKVPoolAllocator,
) -> None:
    from sglang.srt.environ import envs

    # cache_protected_len 必须是 page_size 对齐的
    assert req.cache_protected_len % page_size == 0, "cache_protected_len must be page
    aligned"
    req.swa_evicted_seqlen = max(req.swa_evicted_seqlen, req.cache_protected_len)

    # 减去一页 margin，避免释放达到 radix tree 插入边界，防止 tombstones 导致内存泄漏
```

```
    if envs.SGLANG_OPT_SWA_EVICT_DROP_PAGE_MARGIN.get():
        evict_threshold = pre_len - sliding_window_size
    else:
        evict_threshold = pre_len - sliding_window_size - page_size

    new_swa_evicted_seqlen = max(req.swa_evicted_seqlen, evict_threshold)

    # 按页对齐 evict_seqlen
    if page_size > 1:
        new_swa_evicted_seqlen = (new_swa_evicted_seqlen // page_size) * page_size

    if new_swa_evicted_seqlen > req.swa_evicted_seqlen:
        free_slots = req_to_token_pool.req_to_token[
            req.req_pool_idx, req.swa_evicted_seqlen : new_swa_evicted_seqlen
        ]
        token_to_kv_pool_allocator.free_swa(free_slots)
        req.swa_evicted_seqlen = new_swa_evicted_seqlen
```

python/sglang/srt/mem_cache/unified_cache_components/swa_component.py

实现 free_out_of_window_slots hook，将 SWA 组件特定的参数传递给通用函数，并更新 insert_params。

```
# python/sglang/srt/mem_cache/unified_cache_components/swa_component.py

from sglang.srt.mem_cache.common import free_swa_out_of_window_slots

class SWAComponent(TreeComponent):
    # ... 其他代码 ...

    def free_out_of_window_slots(
        self, req: Req, pre_len: int, insert_params: InsertParams
    ) -> None:
        if self.sliding_window_size is not None:
            free_swa_out_of_window_slots(
                req,
                pre_len,
                sliding_window_size=self.sliding_window_size,
                page_size=self.cache.page_size,
                req_to_token_pool=self.cache.req_to_token_pool,
                token_to_kv_pool_allocator=self.cache.token_to_kv_pool_allocator,
            )
        # 记录 evicted_seqlen 到 insert_params，后续插入时使用
        insert_params.swa_evicted_seqlen = req.swa_evicted_seqlen
```

评论区精华

讨论 1：环境变量 vs CLI 参数 ispobock 建议保持为环境变量，因为该功能目前只对 unified tree 生效，bixue2010 同意并修改。

讨论 2：hook 是否应对所有组件调用 ispobock 询问是否只需要 SWA 组件，bixue2010 认为应作为通用 hook，其他组件（如 Mamba）未来可按需实现。结论：保留对所有组件的循环调用，目前 SWA 实现，其余 no-op。

- 是否应使用 CLI 参数而非环境变量 (design): 改为环境变量 SGLANG_OPT_UNIFIED_CACHE_FREE_OUT_OF_WINDOW_SLOTS。
- hook 是否应对所有组件调用 (design): 保留对所有组件循环调用，目前 SWA 实现释放逻辑，其余组件 no-op。

风险与影响

- 风险：涉及缓存管理逻辑变更，主要风险在于页面边界和对齐条件的正确处理。free_swa_out_of_window_slots 中关于 cache_protected_len 必须 page-aligned 的断言，以及 page_size>1 时的对齐逻辑，若假设不成立可能导致错误释放或内存泄漏。但通过单元测试和环境变量门控（默认关闭），对现有用户无影响。
- 影响：对使用 SWA 的模型（如 DeepSeek 等）有正面影响，减少无效 KV 占用，提升长上下文时的批处理能力。非 SWA 模型无影响。启用环境变量后，每个 chunked prefill 完成后立即释放窗口外槽位，可能略微增加计算开销，但内存收益显著。
- 风险标记：核心缓存路径变更，受环境变量控制，需测试覆盖

关联脉络

- 暂无明显关联 PR