PR #36381 完整报告

sgl-project/sglang

Fix SWA ownership across grouped frees

合并时间：2026-08-26 05:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36381

执行摘要

- 一句话：修复分组释放时 SWA 槽位所有权竞态
- 推荐动作：值得精读，特别是 free_swa 和 free_group_end 的时序调整。

功能与动机

PR 标题与 body 明确指出根本原因：分组释放时延迟解析 full-token ID，缓存协调可能在组刷新前重新映射同一 full-token，导致替换的 SWA 槽位被释放而原始槽位泄漏。该问题与 #36135 相关，后者采用不同的快照屏障策略解决同一底层竞态。

实现拆解

实现分两步：

1. 修改 free_swa 方法：在组内时，先解析当前映射并立即清除，将解析出的 SWA 槽位加入 swa_free_group，而非延迟到组刷新。
2. 修改 free_group_end：直接对 swa_free_group 中的物理槽位调用 swa_attn_allocator.free，不再经过 free_swa 的再次解析。

新增测试 test_free_swa_group_owns_mapping_at_enqueue_time 验证：在组内释放后重新映射同一 full 槽位，最终释放的是原始 SWA 槽位而非替换槽位。

关键文件：

- python/sglang/srt/mem_cache/allocator/swa.py（模块 内存分配器；类别 source；类型 core-logic；符号 free_swa, free_group_end）：核心修复逻辑所在文件
- test/registered/unit/mem_cache/test_swa_unittest.py（模块 SWA 测试；类别 test；类型 test-coverage；符号 test_free_swa_group_owns_mapping_at_enqueue_time）：新增回归测试覆盖所有权竞态场景

关键符号：free_swa, free_group_end, test_free_swa_group_owns_mapping_at_enqueue_time

关键源码片段

python/sglang/srt/mem_cache/allocator/swa.py

核心修复逻辑所在文件

```
# python/sglang/srt/mem_cache/allocator/swa.py
def free_swa(self, free_index: torch.Tensor):
    if free_index.numel() == 0:
        return

    # 先解析映射，确保在当前时刻获取正确的 SWA 槽位
    if self.page_size == 1:
        mapping_indices = free_index
    else:
        mapping_indices = self._expand_to_full_pages(free_index)

    swa_indices = self.full_to_swa_index_mapping[mapping_indices]
    swa_indices = swa_indices[swa_indices > 0]
    # 立即清除映射，避免后续组内重映射导致错误释放
    self.clear_full_to_swa_mapping(mapping_indices)

    if not self.is_not_in_free_group:
        # 组内：暂存解析结果，组刷新时统一释放
        self.swa_free_group.append(swa_indices)
        return

    self.swa_attn_allocator.free(swa_indices)

def free_group_end(self):
    super().free_group_end()
    if self.swa_free_group:
        swa_free_group = self.swa_free_group
        self.swa_free_group = []
        # 直接释放物理槽位，不再经过 free_swa 的再次解析
        self.swa_attn_allocator.free(torch.cat(swa_free_group))
```

test/registered/unit/mem_cache/test_swa_unittest.py

新增回归测试覆盖所有权竞态场景

```
# test/registered/unit/mem_cache/test_swa_unittest.py
def test_free_swa_group_owns_mapping_at_enqueue_time(self):
    _, allocator, _ = _build_swa_tree(
        is_eagle=False,
        kv_size=8,
        kv_size_swa=8,
    )
    old_full = _swa_alloc(allocator, 1)
    new_full = _swa_alloc(allocator, 1)
    assert old_full is not None and new_full is not None
    old_swa = allocator.full_to_swa_index_mapping[old_full].clone()
    new_swa = allocator.full_to_swa_index_mapping[new_full].clone()

    allocator.free_group_begin()
    allocator.free_swa(old_full)
```

```
# 模拟缓存协调在组刷新前重映射同一 full 槽位
allocator.set_full_to_swa_mapping(old_full, new_swa)
allocator.clear_full_to_swa_mapping(new_full)
allocator.free_group_end()

# 验证原始 SWA 槽位被释放，替换槽位保留
torch.testing.assert_close(
    allocator.full_to_swa_index_mapping[old_full], new_swa
)
self.assertTrue(
    torch.isin(old_swa, allocator.swa_attn_allocator.free_pages).item()
)
self.assertFalse(
    torch.isin(new_swa, allocator.swa_attn_allocator.free_pages).item()
)
```

评论区精华

无实质 review 讨论，仅维护者触发重跑测试并批准。

- 暂无高价值评论线程

风险与影响

- 风险：该变更修改了核心内存释放路径，若解析时机不当可能导致释放错误槽位或映射未清除。但已通过新增测试和现有测试验证。
- 影响：影响 SWA 内存分配器的分组释放路径，修复了潜在的槽位泄漏，对使用 SWA 的模型（如滑动窗口注意力）的稳定性和内存利用率有正面影响。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #36135 同一分组释放所有权竞态的另一种修复策略：PR body 提及采用不同的快照屏障策略解决同一底层竞态