PR #27779 完整报告

sgl-project/sglang

Fix paged SWA free mapping cleanup

合并时间：2026-06-11 18:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27779

执行摘要

- 一句话：修复 SWA 分配器分页释放时映射残留导致 double-free
- 推荐动作：建议精读 _expand_to_full_pages 的实现和 free_swa 的变更逻辑，这是一个典型的分配器 bug 修复，体现了分页分配语义与逐 token 映射之间的不匹配问题，设计值得学习。

功能与动机

当 page_size > 1 时，free_swa() 仅清除传入 token 的 full_to_swa_index_mapping，但底层分页分配器以页粒度释放：释放页内任意 token 会释放整个页。如果映射只清除该 token，同页其他 token 仍指向已释放的 SWA 页，后续释放这些 token 时会导致 double-free 并破坏分配器记账。

实现拆解

1. 在 free_swa 中增加空张量提前返回：若 free_index.numel() == 0 直接返回，避免后续操作空张量。
2. 根据 page_size 选择映射索引：page_size == 1 时使用原始 free_index；否则调用 _expand_to_full_pages 将 token 索引扩展为完整页索引数组。
3. 新增 _expand_to_full_pages 辅助方法：通过 indices // page_size 计算唯一页面，再生成页内所有偏移，返回属于这些页的所有 token 索引。
4. 更新映射清除逻辑：使用扩展后的 mapping_indices 替代 free_index 来清除 full_to_swa_index_mapping，确保整个页的映射被置零。
5. 添加回归测试：在 test_swa_unittest.py 中新增 test_swa_memory_pool_paged_free_clears_full_page_mapping，验证分页分配后部分 token 释放能正确恢复所有对应 SWA 空间并清零映射。

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 _expand_to_full_pages）：核心 bug 修复：修改 free_swa 方法，新增 _expand_to_full_pages 辅助函数，确保分页模式下映射清除与分配器语义一致。
- test/registered/unit/mem_cache/test_swa_unittest.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_swa_memory_pool_paged_free_clears_full_page_mapping）：新增回归测试，验证分页分配后部分 token 释放能正确恢复 SWA 空间并清零映射。

关键符号：_expand_to_full_pages, free_swa

关键源码片段

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

核心 bug 修复：修改 free_swa 方法，新增 _expand_to_full_pages 辅助函数，确保分页模式下映射清除与分配器语义一致。

```
    def free_swa(self, free_index: torch.Tensor):
        if free_index.numel() == 0:
            return

        # 根据 page_size 决定使用原始索引还是扩展为整页索引
        if self.page_size == 1:
            mapping_indices = free_index
        else:
            # 扩展为完整页索引，与底层分页分配器语义一致
            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]
        self.swa_attn_allocator.free(swa_indices)
        # 清除整个页的映射，避免同页其他 token 残留失效映射
        self.full_to_swa_index_mapping[mapping_indices] = 0

    def _expand_to_full_pages(self, indices: torch.Tensor) -> torch.Tensor:
        # 计算所有唯一页面
        pages = torch.unique(indices // self.page_size)
        # 生成页内所有偏移 [0, 1, ..., page_size-1]
        page_offsets = torch.arange(
            self.page_size, dtype=indices.dtype, device=indices.device
        )
        # 广播计算出每个页的所有 token 索引，并展平为一维
        return (pages[:, None] * self.page_size + page_offsets[None, :]).reshape(-1)
```

test/registered/unit/mem_cache/test_swa_unittest.py

新增回归测试，验证分页分配后部分 token 释放能正确恢复 SWA 空间并清零映射。

```
    def test_swa_memory_pool_paged_free_clears_full_page_mapping(self):
        page_size = 4
        _, allocator, _ = _build_swa_tree(
            is_eagle=False,
            page_size=page_size,
            kv_size=16,
            kv_size_swa=16,
            sliding_window_size=page_size,
        )

        full_indices = _swa_alloc(allocator, page_size)
```

```
self.assertEqual(allocator.swa_available_size(), 16 - page_size)

# 释放页内第一个 token，应回收整页 SWA 空间
allocator.free_swa(full_indices[:1])
self.assertEqual(allocator.swa_available_size(), 16)
# 验证整页映射均已清零
self.assertTrue(
    torch.all(
        allocator.full_to_swa_index_mapping[full_indices.to(torch.int64)] == 0
    )
)

# 再次释放同页另一个 token，不应 double-free
allocator.free_swa(full_indices[1:2])
self.assertEqual(allocator.swa_available_size(), 16)
```

评论区精华

无实质性讨论。PR 由 ispobock 直接批准，gemini-code-assist[bot] 自动评论未发现问题。ispobock 在 issue 评论中表示该修复解决了资源泄漏问题（关联 issue #27789）。

- 自动代码审查无问题 (other): 无需处理

风险与影响

- 风险：低风险。修改集中在 free_swa 方法内部，新增的 _expand_to_full_pages 只影响 page_size > 1 的分支，page_size == 1 行为不变。测试覆盖了主要场景。潜在风险：如果 full_to_swa_index_mapping 中存在零值索引，_expand_to_full_pages 扩展后可能意外清除非本页映射（但单元测试中 full_indices 均为有效分配索引，未覆盖该边界）。
- 影响：直接影响 SWA 内存分配器的正确性，避免分页模式下的 double-free 和内存泄漏。影响范围限于使用 page_size > 1 的 SWA 分配场景（如 HiCache 相关路径）。对已有 page_size == 1 的部署无影响。
- 风险标记：分配器语义修正，边界情况测试覆盖

关联脉络

- PR #27789 关联 issue: PR 评审者 ispobock 在评论中提及此 issue，表示 PR 修复了该 issue 描述的资源泄漏问题。
- PR #27759 [UnifiedTree]: HybridModel launches HiCache via UnifiedTree by default.: 同为 HiCache 相关变更，涉及 SWA 分配器路径，可能与此 PR 修复的问题有交互。