PR #29351 完整报告

sgl-project/sglang

[bug1] keep full kv when swa skips leaf data

合并时间：2026-07-01 00:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29351

执行摘要

- 一句话：修复 SWA 跳过叶子创建导致 Full KV 丢失的问题
- 推荐动作：值得精读，特别是统一缓存组件设计模式及如何通过消除组件间副作用依赖来保持核心路径的鲁棒性。关注 Review 中关于添加 assert 的讨论，体现了在不同配置边界下的权衡决策。

功能与动机

统一缓存中，Full KV 与辅助组件状态存储在同一 radix node 上，但辅助组件可能对某个跨度没有有效值（例如 SWA 窗口外）。SWA 的叶子创建否决导致 Full KV 无法插入，违背了“叶子仅靠 Full 值存活”的设计原则，引发缓存命中率降低和资源泄漏。

实现拆解

1. 移除叶子创建否决：在 unified_radix_cache.py 的 _insert_helper 中，删除了对所有组件调用 should_skip_leaf_creation 的检查以及相关的 KV 释放逻辑，无论是否存在 tombstone 都直接创建新叶子。
2. 清除相关接口：在 tree_component.py 基类中移除了 should_skip_leaf_creation 方法（返回 False），在 swa_component.py 中移除了具体实现（基于 swa_evicted_seqlen 的判断）。
3. 更新文档：在 tree_component.py 的注释和 README.md 中明确说明叶子仅依赖 Full 值存活，辅助组件的墓碑不应阻止叶子创建。
4. 补充测试：新增 test_swa_insert_keeps_full_leaf_when_entire_span_is_outside_window，验证当整个 span 超出 SWA 窗口时，Full 叶子仍被创建且 Full KV 分配器未泄漏，SWA 值为 None。

关键文件：

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic）：核心修改：移除叶子创建否决，始终物化叶子，保证 Full KV 可缓存。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_swa_insert_keeps_full_leaf_when_entire_span_is_outside_window）：新增测试用例，覆盖整个叶子在 SWA 窗口外时正确行为，确保 Full KV 保留且无泄漏。

- python/sglang/srt/mem_cache/unified_cache_components/tree_component.py（模块 辅助组件；类别 source；类型 core-logic；符号 should_skip_leaf_creation）：移除基类中的 should_skip_leaf_creation 方法，清理接口。
- python/sglang/srt/mem_cache/unified_cache_components/swa_component.py（模块 SWA 组件；类别 source；类型 core-logic；符号 should_skip_leaf_creation）：移除 SWA 组件中的 should_skip_leaf_creation 实现，不再阻止叶子创建。
- python/sglang/srt/mem_cache/unified_cache_components/README.md（模块 文档；类别 docs；类型 documentation）：更新文档，说明叶子创建不再受辅助组件否决。

关键符号：_insert_helper, should_skip_leaf_creation, test_swa_insert_keeps_full_leaf_when_entire_span_is_outside_window

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

核心修改：移除叶子创建否决，始终物化叶子，保证 Full KV 可缓存。

```
# ... 前面的循环逻辑 ...
        is_new_leaf = False
        # 创建剩余后缀的新叶子。叶子仅依靠 Full 值存活；
        # 辅助组件（SWA、Mamba）可能会在此跨度上仅持有墓碑（例如整个叶子在 SWA 窗口外）。
        # 无论如何都要物化叶子，以便 Full KV 保持可缓存。
        if len(key):
            target_node = self._add_new_node(node, key, value, priority=priority)
            is_new_leaf = True
        else:
            target_node = node

        # 最终化：让每个组件将其数据附加到目标节点。
        result = InsertResult(prefix_len=total_prefix_length)
        for component in self._components_tuple:
            component.commit_insert_component_data(
                node=target_node,
                is_new_leaf=is_new_leaf,
                params=params,
                result=result,
            )

        if target_node is not self.root_node:
            for component in self._components_tuple:
                if component.component_type == BASE_COMPONENT_TYPE:
                    continue
                component.refresh_lru(
                    LRURefreshPhase.INSERT_END, target_node, self.root_node
                )

        if is_new_leaf:
            self._inc_hit_count(target_node, params.chunked)
```

```
        return result
```

test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py

新增测试用例，覆盖整个叶子在 SWA 窗口外时正确行为，确保 Full KV 保留且无泄漏。

```
    def test_swa_insert_keeps_full_leaf_when_entire_span_is_outside_window(self):
        # 叶子仅依靠 Full 值存活：即使整个跨度在 SWA 窗口之后（swa_evicted_seqlen >= total），
        # Full 叶子也必须被物化（Full KV 保留），这样前缀才可缓存。
        # 在所有 SWA 配置上运行，包括 page_size > sliding_window_size 的边界情况（dsv4
        风格）。
        if not self.cfg.has_swa or self.cfg.has_mamba:
            self.skipTest("requires SWA without Mamba")
        tree, allocator, _ = build_fixture(self.cfg)

        tokens = self._make_seq(1, 2)
        value = self._alloc(allocator, len(tokens))
        if value is None:
            self.skipTest("insufficient pool for this config")
        full_available_before = allocator.full_attn_allocator.available_size()

        tree.insert(
            InsertParams(
                key=RadixKey(array("q", tokens)),
                value=value,
                prev_prefix_len=0,
                swa_evicted_seqlen=len(tokens),
            )
        )

        # 验证 Full KV 分配器的大小不变（没有泄漏）
        self.assertEqual(
            allocator.full_attn_allocator.available_size(), full_available_before
        )
        node = next(iter(tree.root_node.children.values()))
        # 验证 Full 组件的值就是我们插入的值
        self.assertTrue(
            torch.equal(node.component_data[ComponentType.FULL].value, value)
        )
        # 验证 SWA 组件值为 None（墓碑）
        self.assertIsNone(node.component_data[ComponentType.SWA].value)
        tree.sanity_check()
```

评论区精华

Review 中 Jialin 建议在移除 should_skip_leaf_creation 后添加 assert 以防止完全超出窗口的情况发生，但 ispobock 指出在 page_size > window_size 并启用 SGLANG_OPT_SWA_EVICT_DROP_PAGE_MARGIN 时仍可能出现，且对 cache_finished 场景不危险（仅创建墓碑节点，不会 OOM），故未添加。此外，团队讨论了 should_skip_leaf_creation 的原始设计意图（避免 cache_finished_req 中的墓碑叶子），最

终确认移除该接口是安全的。

- 是否保留 should_skip_leaf_creation 接口？ (design): 最终决定移除该方法，因为叶子创建不再由辅助组件否决，SWA 墓碑不会导致 OOM。
- 是否添加 assert 确保不会出现完全外窗情况？ (correctness): 未添加 assert，保持当前设计。

风险与影响

- 风险:
  1. 兼容性风险：移除了 should_skip_leaf_creation 方法，若其他组件（如未来新增的辅助组件）依赖此接口，需要调整。但当前只有 SWA 使用它，且已通过测试验证。
  2. 内存风险：SWA 墓碑节点可能增多，但 commit_insert_component_data 中的 recover_after_unevict 和 evict 逻辑会正常处理，不会造成内存泄漏。
  3. 性能影响：无显著影响，叶子创建路径略微简化。 - 影响：对用户透明，修复了统一缓存中 Full KV 可能被错误丢弃的问题，提升缓存命中率和推理稳定性。对团队而言，简化了组件 API，减少了一个潜在的不一致代码路径。 - 风险标记：核心路径变更，组件接口删除

关联脉络

- PR #29349 Parent PR (unified cache fix series): 本 PR 从 #29349 拆分，专注于修复 Bug 1（SWA 跳过叶子创建）。