PR #26939 完整报告

sgl-project/sglang

[Bug Fix][HiCache] Drop @lru_cache on UnifiedTreeNode.get_prefix_hash_values

合并时间：2026-06-02 12:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26939

执行摘要

- 一句话：修复 HiCache 统一 radix 树缓存突变
- 推荐动作：值得精读。这是一个典型的多层抽象遗产 bug 修复：统一树从老树复制代码时复制了已被发现有害的装饰器。建议回顾 #26177 和 #26062 的演进历史，理解如何避免类似复制引发的二重 bug。

功能与动机

PR #26062 引入 UnifiedTreeNode.get_prefix_hash_values 时复制了 @lru_cache(maxsize=1) 但未考虑返回的 list[str] 是可变对象。下游 HiCache 存储路径（如 cache_controller._page_transfer/_page_backup）通过 prefix_keys += batch_hashes 就地扩展该列表，导致后续通过同一 (self, node) 键访问缓存时返回已被篡改的列表，造成 hash 重复。详见 #26173 原始报障。

实现拆解

1. 移除装饰器与清理导入：在 python/sglang/srt/mem_cache/unified_radix_cache.py 中删除第 117 行的 @lru_cache(maxsize=1) 装饰器，并从 from functools import lru_cache, partial 改为 from functools import partial，消除未使用导入。
2. 回归测试：在 test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py 中新增 TestUnifiedTreeNodeGetPrefixHashValues 测试类，包含 test_get_prefix_hash_values_not_shared_across_calls 方法。该方法构建一棵 4 节点树，先获取前两个节点的 hash 列表，就地追加模拟下游修改，再次调用同一方法验证返回未修改的原始列表，并确认两个列表对象不同（assertIsNot），同时验证后续更长路径仍能正确返回。
3. 配套改动：无其他文件变更。

关键文件：

- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 bugfix；符号 get_prefix_hash_values）：移除 @lru_cache 装饰器的核心文件，修复可变列表缓存共享导致的正确性问题。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 测试；类别 test；类型 test-coverage；符号 TestUnifiedTreeNodeGetPrefixHashValues, test_get_prefix_hash_values_not_shared_across_calls, make_node）：新增回归测试，

确保 get_prefix_hash_values 返回的列表不被下游就地修改影响。

关键符号：UnifiedTreeNode.get_prefix_hash_values

关键源码片段

python/sglang/srt/mem_cache/unified_radix_cache.py

移除 @lru_cache 装饰器的核心文件，修复可变列表缓存共享导致的正确性问题。

```
# 修复前：@lru_cache(maxsize=1) 缓存了返回的可变 list，下游就地修改后
# 后续调用会读到被篡改的列表，导致 hash 重复。
# 修复后：每次调用都重新计算，避免共享可变对象。
def get_prefix_hash_values(self, node: UnifiedTreeNode) -> list[str]:
    # 递归收集从 node 到根的 hash 列表
    if node is None or node.hash_value is None:
        return []
    # 递归调用 + 当前节点 hash，每次新建 list
    return node.get_prefix_hash_values(node.parent) + node.hash_value
```

test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py

新增回归测试，确保 get_prefix_hash_values 返回的列表不被下游就地修改影响。

```
class TestUnifiedTreeNodeGetPrefixHashValues(CustomTestCase):
    def test_get_prefix_hash_values_not_shared_across_calls(self):
        """Regression guard for cached mutable prefix hash lists (#26177)."""

        def make_node():
            # 创建统一树节点，仅使用 FULL 组件以简化测试
            return UnifiedTreeNode(tree_components=(ComponentType.FULL,))

        # 构建树：root -> n1(h1) -> n2(h2) -> n3(h3)
        root = make_node()
        n1 = make_node()
        n1.parent = root
        n1.hash_value = ["h1"]
        n2 = make_node()
        n2.parent = n1
        n2.hash_value = ["h2"]
        n3 = make_node()
        n3.parent = n2
        n3.hash_value = ["h3"]

        # 从 n3 向上到 n2 的 hash: [h1, h2]
        first = n3.get_prefix_hash_values(n2)
        self.assertEqual(first, ["h1", "h2"])

        # 模拟下游存储代码就地扩展 prefix_keys
        first += ["h3"]

        # 再次调用，应返回未修改的原始列表
```

```
        second = n3.get_prefix_hash_values(n2)
        self.assertEqual(second, ["h1", "h2"])
        self.assertIsNot(second, first)

        # 验证更长路径也能正确工作
        n4 = make_node()
        n4.parent = n3
        n4.hash_value = ["h4"]
        self.assertEqual(n4.get_prefix_hash_values(n3), ["h1", "h2", "h3"])
```

评论区精华

无需讨论，PR 提交者 vuuihc 直接 CC 维护者 hzh0425 指出这是 #26177 在统一树上的对应修复，hzh0425 在没有任何评论的情况下直接批准并合并。

- 暂无高价值评论线程

风险与影响

- 风险：极低风险。移除的仅是一个 maxsize=1 的 LRU 缓存，其命中窗口极窄（仅单节点扩展短列表），性能影响可忽略。返回的 list 不再被缓存共享，下游就地修改也不会污染后续调用，正确性得到保证。
- 影响：影响 HiCache 统一 radix 树分支的 KV 缓存前缀哈希计算，确保哈希列表在多轮操作中保持正确，防止因重复哈希导致缓存匹配异常。影响范围限定于使用 UnifiedRadixCache 的场景（L3 HiStorage）。
- 风险标记：无风险，只移除缓存

关联脉络

- PR #26177 TreeNode.get_prefix_hash_values @lru_cache can return mutated list: 原始修复，在 legacy 树上删除了相同的装饰器，本 PR 是其统一树副本。
- PR #26062 L3 HiStorage support: 引入 UnifiedTreeNode.get_prefix_hash_values 并复制了有问题的 @lru_cache。