PR #26177 完整报告

sgl-project/sglang

[Bug Fix][HiCache] TreeNode.get_prefix_hash_values @lru_cache can return mutated list

合并时间：2026-05-25 11:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26177

执行摘要

- 一句话：修复 HiCache 缓存 list 被下游就地修改的 bug
- 推荐动作：建议合并。该 PR 修复了一个典型的可变对象缓存 alias bug，修改简洁正确，测试覆盖充分。值得精读以理解 @lru_cache 对可变返回值的安全使用要点。

功能与动机

Issue #26173 报告 TreeNode.get_prefix_hash_values 上的 @lru_cache(maxsize=1) 装饰器缓存了可变 List[str]，下游代码如 cache_controller._page_transfer 会就地 extend 该列表，导致后续通过递归调用相同 (self, node) 参数的调用返回污染后的前缀哈希列表，进而引发 HiCache 存储路径 prefix_keys 出现重复或错乱的哈希值。

实现拆解

1. 移除 radix_cache.py 中的 @lru_cache：删除 TreeNode.get_prefix_hash_values 上的 @lru_cache(maxsize=1) 装饰器以及 from functools import lru_cache 导入。
2. 移除 mamba_radix_cache.py 中的 @lru_cache：对 mamba 侧的 TreeNode 做相同操作。
3. 添加回归测试：在 test/registered/unit/mem_cache/test_radix_cache_unit.py 中新增 test_get_prefix_hash_values_not_shared_across_calls，使用 self.subTest 分别测试 radix_cache.TreeNode 和 mamba_radix_cache.TreeNode，通过模拟下游就地修改来验证每次调用返回新列表。

关键文件：

- python/sglang/srt/mem_cache/radix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring）：核心缓存文件，移除 @lru_cache 和 import，确保 get_prefix_hash_values 每次返回新列表。
- python/sglang/srt/mem_cache/mamba_radix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring）：mamba 侧缓存文件，与 radix_cache.py 同步修改，保持一致。
- test/registered/unit/mem_cache/test_radix_cache_unit.py（模块 测试；类别 test；类型 test-coverage；符号 test_get_prefix_hash_values_not_shared_across_calls）：新增回归测试，验证两条路径在就地修改后仍返回正确结果。

关键符号：get_prefix_hash_values, test_get_prefix_hash_values_not_shared_across_calls

关键源码片段

python/sglang/srt/mem_cache/radix_cache.py

核心缓存文件，移除 @lru_cache 和 import，确保 get_prefix_hash_values 每次返回新列表。

```
# python/sglang/srt/mem_cache/radix_cache.py (head)
# 移除了 @lru_cache，每次调用返回新的列表，避免下游就地修改污染缓存
from typing import TYPE_CHECKING, Any, Iterator, List, Optional, Tuple, Union
# from functools import lru_cache # 已删除

class TreeNode:
    # ...
    def get_prefix_hash_values(self, node: TreeNode) -> List[str]:
        if node is None or node.hash_value is None:
            return []
        # 递归拼接，+ 运算符创建新列表，不与旧列表共享引用
        return node.get_prefix_hash_values(node.parent) + node.hash_value
```

python/sglang/srt/mem_cache/mamba_radix_cache.py

mamba 侧缓存文件，与 radix_cache.py 同步修改，保持一致。

```
# python/sglang/srt/mem_cache/mamba_radix_cache.py (head)
# 与 radix_cache.py 保持一致，移除 @lru_cache
from typing import TYPE_CHECKING, List, Optional, Tuple
# from functools import lru_cache # 已删除

class TreeNode:
    # ...
    def get_prefix_hash_values(self, node: "TreeNode") -> List[str]:
        if node is None or node.hash_value is None:
            return []
        return node.get_prefix_hash_values(node.parent) + node.hash_value
```

test/registered/unit/mem_cache/test_radix_cache_unit.py

新增回归测试，验证两条路径在就地修改后仍返回正确结果。

```
# test/registered/unit/mem_cache/test_radix_cache_unit.py (head)
from sglang.srt.mem_cache.mamba_radix_cache import TreeNode as MambaTreeNode
from sglang.srt.mem_cache.radix_cache import TreeNode

class TestRadixTree(unittest.TestCase):
    # ...
    def test_get_prefix_hash_values_not_shared_across_calls(self):
        """Regression guard for cached mutable prefix hash lists."""
        for node_cls in (TreeNode, MambaTreeNode):
            with self.subTest(node_cls=node_cls.__module__):
                root = node_cls()
                n1 = node_cls()
                n1.parent = root
                n1.hash_value = ["h1"]
                n2 = node_cls()
```

```
n2.parent = n1
n2.hash_value = ["h2"]
n3 = node_cls()
n3.parent = n2
n3.hash_value = ["h3"]

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

# 模拟下游代码就地修改 prefix_keys
first += ["h3"]

second = n3.get_prefix_hash_values(n2)
self.assertEqual(second, ["h1", "h2"])
self.assertIsNot(second, first) # 必须是新对象

n4 = node_cls()
n4.parent = n3
n4.hash_value = ["h4"]
# 递归调用不应受之前修改影响
self.assertEqual(
    n4.get_prefix_hash_values(n3), ["h1", "h2", "h3"]
)
```

评论区精华

无实质讨论。gemini-code-assist 的自动 code review 确认了修改意图和正确性；hzh0425 直接 Approved。

- 自动 code review 反馈 (correctness): 无额外反馈，自动评价正确。

风险与影响

- 风险：移除缓存可能导致在短时间内重复调用 get_prefix_hash_values(相同 node) 时增加少量计算开销（递归拼接哈希列表）。但原始 maxsize=1 仅在线性前缀扩展时才有收益，且下游存储路径涉及 L3 I/O，计算开销可忽略。无回归风险，测试已覆盖两条代码路径。
- 影响：影响范围限于启用 HiCache 且 hicache_storage_pass_prefix_keys 为 True 的场景。修复后，prefix_keys 不会再因缓存别名而包含重复或错乱的哈希值，提升 HiCache 存储正确性。其余路径无影响。
- 风险标记：轻微性能回退（可忽略）

关联脉络

- PR #25065 [UnifiedTree] fix: backup SWA-split parent before child under write-through: 同为缓存层 bugfix，修改了 radix_cache.py 相关文件。