

PR #38577 完整报告

sgl-project/sglang

[HiCache][LoRA] Isolate storage pages by extra key

合并时间: 2026-09-12 18:18

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38577>

执行摘要

- 一句话: HiCache 存储页按 LoRA 与 salt 隔离, 修复 KV 跨适配器复用
- 推荐动作: 值得精读。核心看点: ① 跨语言 (Python/Rust) 哈希种子的 framing 编码设计, 如何用长度前缀消除歧义拼接并保持逐字节一致; ② 存储哈希与事件哈希分离的契约边界, 如何在修复正确性的同时不破坏既有外部消费者; ③ root-only 匹配下 namespace 透传的坑位 (last_host_node.key 取不到请求 namespace 的问题)。对涉及多级缓存、跨语言实现的团队有直接借鉴价值。

功能与动机

PR body 明确指出: HiCache 的 L3 storage 哈希只用 token, 导致相同 prompt 在不同 LoRA adapter 或 cache_salt 之间复用 KV 页面, 而不同 adapter 的 KV 内容并不相同, 复用会造成静默错误。reviewer alphabetc1 在评审中进一步强调: 直接把 extra_key 混入事件哈希会破坏既有消费者, 要求复用 #37058 的 storage/event 哈希分离方案——存储哈希带命名空间用于隔离, 事件哈希保持既有 salt 规则用于对外发布。

实现拆解

实现按以下 5 步展开:

1. Python 侧存储哈希种子 (核心): 在 `python/sglang/srt/mem_cache/utils.py` 新增 `storage_namespace_seed` 与 `get_storage_hash_str`。种子用 `b"sglang-cache-namespace-v1"` 前缀 + 逐字段 framing (None 写 `\x00`, 存在字段写 `\x01 + u64 little-endian 长度 + UTF-8 内容`), 长度前缀消除 ("`a`", "`bc`") 与 ("`ab`", "`c`") 的拼接歧义; `compute_node_hash_values` 改走 `get_storage_hash_str`, 链首以命名空间播种、子节点沿用父链最后哈希不重复播种。同时重写 `compute_node_event_hash_values`, 事件哈希只保留 `cache_salt` 旧种子 (`sglang-cache-salt-v1\0` 前缀), `extra_key` 不进入事件链。
2. Rust 侧逐字节对齐: `rust/sglang-radix-tree/src/node.rs` 的 `NodeArena::compute_node_hash_values` 在无父哈希时按同一 framing 计算命名空间种子 (`Sha256 + 0x01 + u64::to_le_bytes`), 保证 Python/Rust 两棵树产出的存储哈希一致; Rust 端新增 `storage_hashes_match_python` 测试锁定字节级兼容。
3. Rust 统一树核心事件哈希改造: `rust/sglang-radix-tree/src/unified_tree_core.rs` 将 `salted_event_hashes` 更名为 `namespaced_event_hashes`, 触发条件从“仅 `cache_salt` 存

在”放宽为“namespace 非默认”，但 `ensure_namespaced_event_hashes_` 的链首种子仍只在 `cache_salt` 存在时生成，使带 `extra_key` 的节点发布事件时保持 token-only 哈希；节点拆分、叶子删除与自检逻辑同步适配新字段。

4. 全链路 namespace 透传：python/sglang/srt/mem_cache/hiradix_cache.py 的 `query_storage_hit_length` 与 `prefetch_from_storage` 改为显式接收 `extra_key/cache_salt` 参数，修复 root-only 匹配时 `last_host_node.key` 是默认 namespace 导致取不到请求命名空间的问题；python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py 的 `_compute_prefix_hash` 增加 `extra_key/cache_salt` 参数并在链首用 `storage_namespace_seed` 播种，使 decode 侧 offload 的哈希链与 prefill 侧保持一致。unified_radix_cache.py、unified_cache_linker.py、events.py、cache_controller.py、hybrid_cache_controller.py、decode_hicache_mixin.py 等为配套接线改动。
5. 测试与验证配套：新增端到端测试 test/registered/e2e/hicache/test_hicache_storage_lora.py（file 后端 + LoRA 真实服务，验证 adapter/base/salt 三向隔离与 storage 往返命中）；test_mem_cache_utils.py 新增 TestStorageHashNamespace 覆盖 6 种命名空间组合的隔离与链延续、歧义拼接；test_radix_cache_unit.py 新增 test_extra_key_does_not_move_published_block_hashes 验证事件哈希不受 extra_key 影响；test_specv2_kvcache_offloading.py 验证 decode offload 分块哈希与整体哈希一致；Rust 侧 tests/unified_tree_core.rs 覆盖命名空间事件哈希的稀疏性、节点拆分延续与 token-only 事件发布。

关键文件：

- rust/sglang-radix-tree/src/node.rs（模块 哈希链；类别 source；类型 core-logic；符号 compute_node_hash_values）：Rust 树核心的存储哈希种子实现，与 Python 侧逐字节对齐的关键文件；决定 Rust 树产出的 L3 页面哈希是否隔离命名空间。
- python/sglang/srt/mem_cache/utils.py（模块 哈希工具；类别 source；类型 core-logic；符号 storage_namespace_seed, get_storage_hash_str, compute_node_hash_values, compute_node_event_hash_values）：Python 侧哈希核心工具，新增命名空间种子并分离存储哈希与事件哈希两条链，是全部 Python 路径的行为源头。
- rust/sglang-radix-tree/src/unified_tree_core.rs（模块 事件哈希；类别 source；类型 core-logic；符号 ensure_salted_event_hashes_, ensure_namespaced_event_hashes_）：Rust 统一树核心的事件哈希改造：salted_event_hashes 更名为 namespaced_event_hashes，触发条件扩展但种子仍只用 cache_salt，保证 extra_key 节点发布 token-only 事件哈希。
- test/registered/e2e/hicache/test_hicache_storage_lora.py（模块 集成测试；类别 test；类型 test-coverage；符号 TestHiCacheStorageLoRAIsolation, setUpClass, tearDownClass, send_request）：新增端到端回归测试，真实启动 file 后端 + LoRA 服务，验证 adapter/base/salt 三向隔离、storage 往返命中与文本一致性，是本 PR 最具说服力的正确性证据。
- python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py（模块 解码卸载；类别 source；类型 core-logic；符号 _compute_prefix_hash, _trigger_backup, offload_kv_cache）：decode 侧 offload 哈希链与 prefill 对齐的关键改动；

_compute_prefix_hash 增加命名空间种子，PD 分离场景下 decode 才能命中 prefill 写入的 L3 页面。

- python/sglang/srt/mem_cache/hiradix_cache.py (模块 存储预取; 类别 source; 类型 core-logic; 符号 query_storage_hit_length, prefetch_from_storage) : 修复 root-only 匹配时命名空间取不到的问题: query_storage_hit_length 与 prefetch_from_storage 不再依赖 last_host_node.key, 改为显式接收请求的 extra_key/cache_salt。
- rust/sglang-radix-tree/src/tests/unified_tree_core.rs (模块 核心测试; 类别 test; 类型 test-coverage; 符号 salted_event_hashes_are_sparse_and_removed_with_the_node, namespaced_event_hashes_are_sparse_and_removed_with_the_node, salted_event_hashes_survive_node_split, extra_key_nodes_publish_token_only_event_hashes) : Rust 侧覆盖命名空间事件哈希的稀疏性、节点删除清理、节点拆分延续与 extra_key 节点发布 token-only 事件, 是跨语言一致性的核心保障。
- test/registered/unit/mem_cache/test_mem_cache_utils.py (模块 哈希测试; 类别 test; 类型 test-coverage; 符号 init, TestStorageHashNamespace, test_node_hashes_isolate_namespaces_and_continue_the_chain, child) : 验证命名空间种子的隔离性、链延续与歧义拼接防护, 覆盖 6 种 namespace 组合与父链续接逻辑, 是 Python 哈希行为最直接的单元保障。
- test/registered/unit/disaggregation/test_specv2_kvcache_offloading.py (模块 卸载测试; 类别 test; 类型 test-coverage; 符号 test_decode_offload_hash_chain_matches_prefill) : 验证 decode offload 分块哈希链与 prefill 整体哈希一致, 是 PD 分离场景下命名空间透传正确性的关键回归测试。
- test/registered/unit/mem_cache/test_radix_cache_unit.py (模块 缓存单测; 类别 test; 类型 test-coverage; 符号 test_extra_key_does_not_move_published_block_hashes) : 验证事件哈希不随 extra_key 变化, 确保对外发布的 KV 事件哈希兼容性, 直接对应 review 中 alphabetc1 的关切。

关键符号: storage_namespace_seed, get_storage_hash_str, compute_node_hash_values, compute_node_event_hash_values, ensure_namespaced_event_hashes_, NodeArena::compute_node_hash_values, _compute_prefix_hash, query_storage_hit_length, prefetch_from_storage

关键源码片段

rust/sglang-radix-tree/src/node.rs

Rust 树核心的存储哈希种子实现, 与 Python 侧逐字节对齐的关键文件; 决定 Rust 树产出的 L3 页面哈希是否隔离命名空间。

```
/// 计算节点每页哈希: 优先沿父链延续, 链首则用命名空间播种 (与 Python 逐字节一致) 。
pub fn compute_node_hash_values(&self, node_id: NodeIdx, page_size: usize) -> Vec<String> {
    let node = self.node(node_id);
    let parent_hash = node.parent.and_then(|parent_id| {
        let parent = self.node(parent_id);
        parent.hash_value.as_ref().and_then(|hashes| hashes.last().cloned())
    });
}
```

```

// 仅链首需要播种：父链已存在时直接沿用，避免命名空间被重复混入中间页。
let prior = parent_hash.map(str::to_owned).or_else(|| {
    let namespace = node.namespace.as_ref();
    if namespace == KeyNamespaceRef::default() {
        return None; // 无命名空间：保持纯 token 哈希，向后兼容。
    }
    // 与 Python `storage_namespace_seed` 的 framing 完全一致：
    // 0x01 存在标记 + u64 little-endian 长度 + UTF-8 字节，None 写 0x00。
    let mut hasher = Sha256::new();
    hasher.update(b"sglang-cache-namespace-v1");
    for part in [namespace.extra_key, namespace.cache_salt] {
        match part {
            None => hasher.update([0u8]),
            Some(part) => {
                hasher.update([1u8]);
                hasher.update((part.len() as u64).to_le_bytes());
                hasher.update(part.as_bytes());
            }
        }
    }
    Some(digest_to_hex(&hasher.finalize().into()))
});

crate::node::get_hash_str::<K>(node.key.as_ref(), prior.as_deref(), page_size)
}

```

test/registered/e2e/hicache/test_hicache_storage_lora.py

新增端到端回归测试，真实启动 file 后端 + LoRA 服务，验证 adapter/base/salt 三向隔离、storage 往返命中与文本一致性，是本 PR 最具说服力的正确性证据。

```

def test_adapter_pages_are_isolated_in_storage(self):
    prompt = self.gen_prompt(PROMPT_TOKENS)
    hit_floor = PROMPT_TOKENS - 2 * PAGE_SIZE

    # 冷启动：adapter 请求把页面写入 device、host 与 storage 三层。
    lora_first = self.send_request(prompt, lora_path=LORA_NAME)
    self.flush_device_cache()

    # flush 后再次请求必须从 storage 命中，证明 adapter 页面已持久化。
    lora_again = self.send_request(prompt, lora_path=LORA_NAME)
    self.assertGreater(
        self.cached_tokens(lora_again),
        hit_floor,
        "the adapter's pages were not served from storage after the flush",
    )
    self.assertEqual(lora_first["text"], lora_again["text"])
    self.flush_device_cache()

    # 关键断言：base 与不同 salt 的请求不得复用 adapter 已存的页面。

```

```
# 首次请求缓存命中数必须小于一页，即对既有 namespace 完全 miss。
for cache_salt in (None, "tenant-a"):
    with self.subTest(cache_salt=cache_salt):
        first = self.send_request(prompt, lora_path=None, cache_salt=cache_salt)
        self.assertLess(self.cached_tokens(first), PAGE_SIZE)
        self.flush_device_cache()

        # 同 namespace 的二次请求可从 storage 命中，且输出文本一致。
        again = self.send_request(prompt, lora_path=None, cache_salt=cache_salt)
        self.assertGreater(self.cached_tokens(again), hit_floor)
        self.assertEqual(first["text"], again["text"])
        self.flush_device_cache()

# adapter 页面不受上述 base/salt 请求影响，仍可从 storage 命中。
lora_third = self.send_request(prompt, lora_path=LORA_NAME)
self.assertGreater(self.cached_tokens(lora_third), hit_floor)
self.assertEqual(lora_first["text"], lora_third["text"])
```

评论区精华

评审中最有价值的交锋来自 alphabetc1 (#37058 作者)：

- 事件哈希兼容性（已解决）：alphabetc1 在 REVIEWED 评论中指出“This also changes unsalted KV-event hashes for requests with extra_key, breaking consumers using the existing hash rules”，要求复用 #37058 的分离方案、让 extra_key 不进事件哈希。作者在第二个 commit 中采纳，事件哈希只由 cache_salt 播种。
- 整合 salt 支持（已解决）：alphabetc1 在 Issue 评论中询问“Would you be open to incorporating that here?”，作者将 #37058 的 salt 变更直接 carry 进本 PR，alphabetc1 成为共同作者（commit 信息含 Co-authored-by）。
- 非阻塞 API 建议（未采纳）：alphabetc1 在 `decode_kvcache_offload_manager.py` 的两条行内评论建议“pass req into `_compute_prefix_hash`”和“`storage_namespace_seed(req)`？”，作者保持显式传参风格，评审人标注 not a blocker，最终双 APPROVED。
 - extra_key 进入事件哈希破坏既有消费者 (correctness): 作者在第二个 commit 中采纳：事件哈希只由 cache_salt 播种，extra_key 不参与；Rust 端 `ensure_namespaced_event_hashes_` 与 Python 端 `compute_node_event_hash_values` 同步调整，并新增 token-only 事件测试锁定。
 - 整合 #37058 的 cache_salt 支持 (design): 作者接受整合，第二个 commit 将 salt 与 extra_key 统一 framing 进 storage 种子，alphabetc1 成为共同作者，避免两条并行实现冲突。
 - `_compute_prefix_hash` 签名设计 (style): 作者保持显式 extra_key/cache_salt 参数风格，评审人未阻塞，最终 APPROVED。

风险与影响

- 风险：

1. 跨语言双实现漂移风险: Rust node.rs 与 Python utils.py 各维护一份 framing 编码, 虽有 storage_hashes_match_python 与 36 个 namespace 编码用例锁定, 但后续任何一侧改动都需同步另一侧。
2. 事件哈希内部触发面扩大: ensure_namespaced_event_hashes_ 的触发条件从“仅 cache_salt”扩展到“namespace 非默认”, 带 extra_key 的请求现在也会生成事件哈希 (值本身仍 token-only), 内部状态变更需要既有事件测试持续覆盖。
3. decode offload 断链风险: _compute_prefix_hash 中 prior_hash or storage_namespace_seed(...) 依赖调用方传参正确, 若 prefill 侧未传 namespace 而 decode 侧传入, 首块哈希会不连续; 测试覆盖了 4 种 namespace 组合, 但真实 PD 分离场景仍需观察。
4. GPU 实测覆盖不足: PR body 明确说明 rebase 与 salt 扩展后未在 GPU 上重跑精度测试 (GB300 数据基于旧 revision), CI 各矩阵通过但 AMD ROCm 有一条失败记录。
- 影响: 用户侧: 修复了 LoRA 多适配器与 cache_salt 多租户场景下 HiCache L3 存储页被错误复用的静默 KV 错误; 无命名空间的既有请求哈希完全不变, 向后兼容。系统侧: 统一缓存树 (Python UnifiedRadixCache 与 Rust UnifiedTreeCore) 的存储哈希语义升级为命名空间感知, probe/prefetch/offload 全链路同步, LoRA 场景下 storage 命中率因隔离而下降属预期代价。团队侧: 确立了“存储哈希带命名空间、事件哈希只带 salt”的契约, 并以此方式合入 #37058, 避免后续分支冲突。- 风险标记: 核心路径变更, 跨语言一致性要求, 事件哈希兼容性保护, GPU 复测缺失, 双实现漂移风险

关联脉络

- PR #37058 [HiCache] Add cache_salt namespace support (by alphabetc1): 本 PR 在第二个 commit 中直接 carry 了 #37058 的 cache_salt 支持, 将其与 extra_key 统一 framing 进 storage 哈希种子, 并在 review 中采纳其 storage/event 哈希分离建议。
- PR #37584 [Unified Tree] Port SWA Branching-Point Caching to the Rust TreeCore: 同一 rust/sglang-radix-tree 统一树核心演进线, 本 PR 对 unified_tree_core.rs 事件哈希与节点拆分逻辑的改动与之相邻, 共同推动 Rust TreeCore 能力补齐。