PR #25088 完整报告

sgl-project/sglang

[UnifiedRadixCache] Fix HiCache load back start node

合并时间：2026-05-14 13:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25088

执行摘要

- 一句话：统一 HiCache L2 load-back 锚点到 best_match_node，修复高并发 worker crash
- 推荐动作：该 PR 是核心 bugfix，强烈建议所有启用 HiCache 的用户升级。值得精读源码中的 Full 组件锁跳过 evicted 段的设计模式，以及 SWA 组件如何利用 best_match_node 保证 walk 不越界。测试用例的 setup 函数也展示了复杂的树结构构建技巧，对理解 HiCache 测试有帮助。

功能与动机

Issue #24869 报告高并发下 Gemma 4 模型 worker crash，stack trace 显示在 swa_component.py 的 build_hicache_transfers 中 assert(cd.host_value is not None or cd.value is not None) 失败。根本原因是 load-back 起始节点 last_host_node 可能指向一个不受 component validator 保护的节点，导致加载到无效节点。PR body 明确统一 L2 load-back 锚点到 best_match_node。

实现拆解

1. 数据结构扩展：在 base_prefix_cache.py 中为 MatchResult 新增 best_match_node 字段，并用 InitLoadBackParams.best_match_node 替换原来的 last_host_node。同时在 zero_match_result 中填充默认值。
2. 核心匹配逻辑更新：修改 unified_radix_cache.py 的 _match_prefix_helper 和 _match_prefix_helper_readonly，将内部变量重名为 best_match_node，并在返回时传递给 _match_post_processor，最终构造 MatchResult 时填充该字段。
3. 组件锚点迁移：在 SWA 组件的 finalize_match_result 和 build_hicache_transfers 中使用 result.best_match_node 替代 result.last_host_node 作为 walk 起点。Mamba 组件做同样修改。Full 组件的 build_hicache_transfers 中强化 assert 保证 host_value 非空。
4. Full 组件锁修复：在 full_component.py 的 acquire_component_lock 中新增跳过 evicted 段逻辑：从起始节点向上遍历直到找到第一个有 device value 的节点，记录跳过节点到 skip_lock_node_ids；release_component_lock 相应忽略这些节点。这修复了 write-through 策略下锁定已驱逐节点的问题。
5. 调度与兼容性适配：在 schedule_policy.py 和 schedule_batch.py 中传递 best_match_node 字段。在 9 个遗留缓存（radix_cache、mamba_radix_cache、

hiradix_cache、hi_mamba_radix_cache 等）的 zero_match_result 和 empty_match_result 中填充 best_match_node，确保向后兼容。

6. 测试配套：新增 4 个单元测试覆盖最佳匹配节点在 match_prefix、load_back、finalize 中的行为，并重构辅助函数 _swa_anchor_setup 和 _swa_anchor_chain_tokens。CI 全部通过。

关键文件：

- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 测试；类别 test；类型 test-coverage；符号 _swa_anchor_chain_tokens, _swa_anchor_setup, test_hicache_swa_match_prefix_picks_best_match_node_above_last_host, test_hicache_swa_load_back_anchored_on_best_match_node）：新增 4 个测试用例验证最佳匹配节点行为，重构辅助函数，确保修复稳定性
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 核心缓存；类别 source；类型 core-logic；符号 _match_prefix_helper, _match_post_processor, match_prefix, _reset_full）：核心缓存入口，匹配和插入主路径，best_match_node 的生成与传递
- python/sglang/srt/mem_cache/unified_cache_components/full_component.py（模块 全组件；类别 source；类型 core-logic；符号 acquire_component_lock, release_component_lock, build_hicache_transfers）：Full 组件锁跳过 evicted 分段，修复锁定计数问题
- 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 finalize_match_result, build_hicache_transfers）：SWA 组件使用 best_match_node 锚点，保证 load-back 不越过滑动窗口重置点
- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 数据结构；类别 source；类型 data-contract；符号 MatchResult, InitLoadBackParams, zero_match_result）：数据结构定义，新增 best_match_node 字段，所有缓存的基础
- python/sglang/srt/mem_cache/hi_mamba_radix_cache.py（模块 兼容层；类别 source；类型 compatibility；符号 init_load_back, match_prefix, _match_post_processor）：兼容性填充，确保 Mamba 专用缓存正确初始化 best_match_node

关键符号：_match_prefix_helper, _match_post_processor, acquire_component_lock, release_component_lock, finalize_match_result, build_hicache_transfers, zero_match_result, init_load_back

关键源码片段

python/sglang/srt/mem_cache/unified_cache_components/full_component.py

Full 组件锁跳过 evicted 分段，修复锁定计数问题

```
# python/sglang/srt/mem_cache/unified_cache_components/full_component.py

def acquire_component_lock(
    self, node: UnifiedTreeNode, result: IncLockRefResult
```

```
) -> IncLockRefResult:
    ct = self.component_type
    root = self.cache.root_node
    cur = node
    # 跳过底部已驱逐的 evicted 段 (value is None)
    while cur is not root and cur.component_data[ct].value is None:
        result.skip_lock_node_ids.setdefault(ct, set()).add(cur.id)
        cur = cur.parent
    # 锁定设备上的段 ( 从第一个有 device value 的节点到 root)
    delta = 0
    while cur is not root:
        cd = cur.component_data[ct]
        assert cd.value is not None, (
            f'FULL invariant broken: evicted ancestor {cur.id} above device-on segment'
        )
        if cd.lock_ref == 0:
            key_len = len(cd.value)
            self.cache.component_evictable_size_[ct] -= key_len
            self.cache.component_protected_size_[ct] += key_len
            delta += key_len
        cd.lock_ref += 1
        self.cache.evictable_device_leaves.discard(cur)
        cur = cur.parent
    result.delta = delta
    return result
```

python/sglang/srt/mem_cache/unified_cache_components/swa_component.py

SWA 组件使用 best_match_node 锚点，保证 load-back 不越过滑动窗口重置点

```
# python/sglang/srt/mem_cache/unified_cache_components/swa_component.py

def finalize_match_result(
    self,
    result: MatchResult,
    params: MatchPrefixParams,
    value_chunks: list[torch.Tensor],
    best_value_len: int,
) -> MatchResult:
    ct = self.component_type
    n_swa = 0
    # 使用 best_match_node 作为起点，确保 walk 不越过 SWA 重置点
    node = result.best_match_node
    root = self.cache.root_node
    while node is not root and n_swa < self.sliding_window_size:
        cd = node.component_data[ct]
        if cd.value is None and cd.host_value is not None:
            # host-only tombstone: 标记 host_hit_length
            return result._replace(host_hit_length=max(result.host_hit_length, 1))
```

```
            if cd.value is not None:
                n_swa += len(cd.value)
            elif cd.host_value is not None:
                n_swa += len(cd.host_value)
            else:
                break
            node = node.parent
        return result
```

python/sglang/srt/mem_cache/base_prefix_cache.py

数据结构定义，新增 best_match_node 字段，所有缓存的基础

```
# python/sglang/srt/mem_cache/base_prefix_cache.py

@dataclasses.dataclass
class InitLoadBackParams:
    '''Unified parameters for init_load_back across different cache types.'''
    best_match_node: Any # 锚点：所有组件验证器接受的最深节点，用于 L2 load-back 起始
    host_hit_length: int
    mem_quota: Optional[int] = None
    req: Optional[Req] = None

class MatchResult(NamedTuple):
    '''Result of prefix matching.'''
    device_indices: torch.Tensor
    last_device_node: Any # 设备上的最后匹配节点
    last_host_node: Any # 主机上的最后匹配节点（保留给 L3 prefetch）
    best_match_node: Any # 新增：所有组件验证器接受的节点，用作 L2 load-back 锚点
    host_hit_length: int = 0
    mamba_branching_seqlen: Optional[int] = None
    cache_protected_len: Optional[int] = None
    skip_lock_node_ids: Optional[Dict[ComponentType, Set[int]]] = None
```

评论区精华

在 review 中，hzh0425 对 full_component.py 的 acquire_component_lock 变更提出疑问，认为该逻辑只对 write-through 策略生效。ispobock 回应指出其他策略也适用，因为 Full 组件有一个不变性：如果子节点有 device value，父节点也必须在 device 上。该解释被接受。此外，Issue 评论中 alphabetc1 询问 mamba_component.py 是否需要同步修改，ispobock 确认需要，并在提交中补齐。

- Full 组件锁跳过 evicted 段仅对 write-through 生效？ (correctness): 经 ispobock 解释，设计一致，讨论已解决。

风险与影响

- 风险：核心路径变更：match_prefix 返回值语义变化，所有调用者统一使用 best_match_node 作为锚点，本次已全面替换但需要进一步验证无遗漏。向后兼容风险：9 个遗留缓存必须填充 best_match_node 字段，已在本次修改中覆盖，但若未来增加新的缓

存类型可能遗漏。功能回归：Full 组件锁跳过 evicted 段可能破坏锁定计数，但根据 invariant 保证安全，且已有测试覆盖临时锁场景。性能影响：几乎无开销，额外遍历 evicted 段仅在锁定阶段进行，且通常只跨越有限节点。

- 影响：用户影响：修复高并发下 worker crash，提高系统稳定性，HiCache 功能得以可靠使用。系统影响：统一锚点后 L2 load-back 更健壮，减少 SWA/Mamba 组件中无效加载的风险，为后续 L3 prefetch 预留 last_host_node 的语义。团队影响：代码可读性提升，字段命名更加符合实际语义，降低后续维护难度。但需要确保所有组件开发者遵循统一锚点约定。
- 风险标记：核心路径变更，向后兼容风险，多组件同步

关联脉络

- 暂无明显关联 PR