PR #31902 完整报告

sgl-project/sglang

[UnifiedTree] fix: drop prefetched host refill under an un-backed-up parent

合并时间：2026-07-27 13:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31902

执行摘要

- 一句话：修复 HiCache 预取挂载破坏 backup 不变量导致的调度器崩溃
- 推荐动作：值得精读。亮点在于： 1) 对“两个崩溃症状、一个根因”的剖析方式非常清晰，可作为缓存不变量类 bug 的排查范式； 2) “简单丢弃而非修补”的决策展示了最小化复杂度原则——放弃一次 best-effort 优化换取结构不变量的强保证； 3) review 中 write-through/write-back 语义区分提醒读者策略条件的重要性； 4) rebase 中发现的同构非法状态测试（#29901）是理解 radix-tree 不变量边界的绝佳案例。建议同步关注 #31812 解除跳过后回归测试的恢复情况。

功能与动机

PR body 明确指出：HiCache 将每个 radix-tree 节点的 KV 放在 device 与 host 两层，write-through 策略下 host 副本构成从根节点开始的连续链，缓存必须维持“节点被备份当且仅当其父节点被备份”的不变量。预取提交 _insert_helper_host 直接把新 host 子节点挂到 match anchor 下，而 anchor 经常是仅含 device KV、尚未备份的节点，于是产生“父未备份、子已备份”的非法形状，最终在两个位置爆炸：空闲自检直接 abort（Sanity check FAILED: node N backed up but parent M not backed up），以及驱逐时 _evict_device_leaf -> _remove_leaf_from_parent -> assert v == node 崩溃。hzh0425 在 issue 评论中给出了关键建议：“Perhaps it's better to simply drop the prefetch result rather than trying to patch it”，作者据此将方案从“备份父节点”改为“直接丢弃 refill”，避免额外 edge case。

实现拆解

实现按以下 3 个步骤拆解：

1. 数据契约扩展：在 python/sglang/srt/mem_cache/base_prefix_cache.py 的 InsertResult 数据类中新增 host_insert_dropped: bool = False 字段，用于把“预取 refill 被丢弃”这一事件从树内核传递回缓存控制器，这是后续分支处理的通信基础。
2. 核心修复（树内核层）：在 python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py 的 insert_host 中，沿 key 下行匹配并处理拆分后，若仍有剩余 key 需要新建节点，则先检查 node is not self.root_node and not node.backuped and not self.is_write_back——即 anchor 非根、未备份且当前不是 write-back 策略时，记录 info 日志并设置result.host_insert_dropped=True后直接返回，不再创建backed-up子节点。关键点是 not self.is_write_back 限定：write-back 策略没有 backed-up-parent 约束，refill 仍应保留（这是 review 中 hzh0425 明确提出的修正）。

3. 调用方资源回收：在 python/sglang/srt/mem_cache/unified_radix_cache.py 的 check_prefetch_progress 中按 insert_result.host_insert_dropped 分支：若被丢弃，则将 host_indices[:completed_tokens] 的全部已完成 buffer 以及各组件（SWA/Mamba 等）的 comp_xfers 追加进 append_host_mem_release 释放队列，loaded_from_storage 记 0；否则走原有提交路径（commit_hicache_transfers + 部分释放）。日志也从固定 "success" 改成 "dropped"/"success" 二态，并输出 released_tokens。
4. 测试配套（含临时格式化调整）：按 hzh0425 的 review 要求，回归测试直接并入 test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py，新增 TestUnifiedRadixPrefetchCorruption 类，包含 test_prefetch_refill_under_unbacked_parent_is_dropped（断言子节点为 None、父节点 children 为空、sanity_check 通过）与 test_dropped_prefetch_releases_all_host_resources（用 mock 校验 drop 分支调用了包含完整 buffer 与 extra pools 的 append_host_mem_release）。此外补上了若干既有测试的 with 语法格式化（受控流调整波及）以及 insert_result.host_insert_dropped = False 的 mock 默认值。注意：该测试模块当前仍被 pytestmark = skip(...) 与 setUpModule 整体跳过（原因是 ServerArgs 配置命名空间迁移期，见 #31812），回归测试尚未在 CI 中实际执行。

关键文件：

- python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py（模块 缓存树；类别 source；类型 core-logic；符号 insert_host）：核心修复点：insert_host 在挂载新 host 子节点前检查 anchor 是否已备份，write-through 下未备份则设置 host_insert_dropped 并丢弃 refill，从根上避免破坏 backup 不变量。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存控制；类别 source；类型 core-logic；符号 check_prefetch_progress）：调用方处理：check_prefetch_progress 根据 insert_result.host_insert_dropped 分支，被丢弃时把全部已完成 buffer 与各组件 extra transfers 一次性放入 host 释放队列，避免页面泄漏并补齐日志与指标。
- python/sglang/srt/mem_cache/base_prefix_cache.py（模块 缓存协议；类别 source；类型 data-contract；符号 InsertResult）：数据契约：InsertResult 新增 host_insert_dropped 字段，承载树内核到缓存控制器的丢弃事件传递，是整体修复的通信基础。
- 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 TestUnifiedRadixPrefetchCorruption, _init_hicache, _insert_device, _attach_host_child）：回归测试：新增 TestUnifiedRadixPrefetchCorruption 类，覆盖 write-through 下 refill 被丢弃（sanity_check 通过）与丢弃后 host 资源全部释放两个场景；同时包含 hzh0425 补充的 write-back 保留语义测试。注意该模块当前整体被跳过（ServerArgs 迁移期）。

关键符号：UnifiedTreeCore.insert_host, UnifiedRadixCache.check_prefetch_progress, TestUnifiedRadixPrefetchCorruption._init_hicache, TestUnifiedRadixPrefetchCorruption._insert_device, TestUnifiedRadixPrefetchCorruption._attach_host_child, TestUnifiedRadixPrefetchCorruption.test_prefetch_refill_under_unbacked_parent_is_drop

ped, TestUnifiedRadixPrefetchCorruption.test_dropped_prefetch_releases_all_host_resources

关键源码片段

python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py

核心修复点：insert_host 在挂载新 host 子节点前检查 anchor 是否已备份，write-through 下未备份则设置 host_insert_dropped 并丢弃 refill，从根上避免破坏 backup 不变量。

```
# python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py
# insert_host 的尾部：已完成 prefix 匹配，剩余 key 需要作为新 host 子链挂到 anchor 下。

# 先沿 key 下行匹配已有节点，处理部分匹配的节点拆分
while len(key) > 0 and child_key in node.children:
    node = node.children[child_key]
    self._touch_node(node)
    prefix_len = node.key.match(key, page_size=self.page_size)
    key = key[prefix_len:]
    host_value = host_value[prefix_len:]
    hash_value = hash_value[prefix_len // self.page_size :]
    matched_length += prefix_len
    if prefix_len < len(node.key):
        node, action = self._split_node(node.key, node, prefix_len)
        if action is not None:
            cache_actions.append(action)
    if len(key):
        child_key = key.child_key(self.page_size)

result = InsertResult(
    prefix_len=matched_length, total_len=total_len, cache_actions=cache_actions
)
if len(key) == 0:
    # 完全命中已有节点：无需新建，记录命中的 host 节点即可
    if (
        node is not self.root_node
        and node.component_data[BASE_COMPONENT_TYPE].host_value is not None
    ):
        result.inserted_host_node = node.id
    return result

# 关键修复：write-through 下，backed-up 的 host 子节点只能挂在已备份的父节点下。
# 若 anchor 是纯 device 节点（从未备份），直接丢弃本次 best-effort refill，
# 而不是挂出“父未备份、子已备份”的非法链——该形状会让空闲自检 abort，
# 并在驱逐走 _remove_leaf_from_parent 时触发 assert v == node 崩溃。
# 注意 is_write_back 门控：write-back 策略没有 backed-up-parent 约束，不丢弃。
if node is not self.root_node and not node.backuped and not self.is_write_back:
    logger.info(
        "HiCache prefetch dropped %d-token refill under un-backed-up node %d",
        len(host_value),
```

```
            node.id,
        )
        result.host_insert_dropped = True
        return result

    # 正常路径：在 anchor 下创建 backed-up 子节点并维护 evictable leaf 集合
    new_node = self._new_node(priority=node.priority)
    new_node.parent = node
    new_node.key = key
    new_node.hash_value = hash_value
    new_node.component_data[BASE_COMPONENT_TYPE].host_value = host_value.clone()
    node.children[child_key] = new_node
    self._update_evictable_leaf_sets(new_node)
    self._update_evictable_leaf_sets(node)
    result.inserted_host_node = new_node.id
    return result
```

python/sglang/srt/mem_cache/unified_radix_cache.py

调用方处理：check_prefetch_progress 根据 insert_result.host_insert_dropped 分支，被丢弃时把全部已完成 buffer 与各组件 extra transfers 一次性放入 host 释放队列，避免页面泄漏并补齐日志与指标。

```
    # python/sglang/srt/mem_cache/unified_radix_cache.py
    # check_prefetch_progress：预取完成后把 host 侧结果拼接入树。
    # 先应用 host-insert walk 产生的动作（如节点拆分）
    self._apply_cache_actions(insert_result.cache_actions)

    if insert_result.host_insert_dropped:
        # 修复：anchor 未备份（write-through 下）时 refill 被树内核丢弃。
        # 这里负责归还本次预取占用的全部 host 页面：既包括已完成 buffer
        #  （host_indices 前 completed_tokens 个），也包括各组件（SWA/Mamba 等）
        # 的 extra transfers，全部进入异步释放队列，避免泄漏。
        self.cache_controller.append_host_mem_release(
            host_indices=host_indices[:completed_tokens],
            extra_pools=[x for xfers in comp_xfers.values() for x in xfers],
        )
        loaded_from_storage = 0
        released_tokens = completed_tokens
    else:
        # 正常提交路径：提交 transfer 后归还“未使用”的尾部 buffer
        commit_actions: list[CacheAction | ComponentAction] = []
        self.tree_core.commit_hicache_transfers(
            last_host_node_id,
            CacheTransferPhase.PREFETCH,
            comp_xfers,
            cache_actions=commit_actions,
            insert_result=insert_result,
            pool_storage_result=operation.pool_storage_result,
        )
```

```
            self._apply_cache_actions(commit_actions)
            assert not insert_result.cache_actions
            self.cache_controller.mem_pool_host.free(host_indices[: insert_result.prefix_len])
            self.cache_controller.append_host_mem_release(
                host_indices[min_completed_tokens:completed_tokens]
            )
            loaded_from_storage = min_completed_tokens - insert_result.prefix_len
            released_tokens = completed_tokens - min_completed_tokens

        self.dec_host_lock_ref(last_host_node_id, anchor_lock_params)
        del self.ongoing_prefetch[req_id]
        self.cache_controller.prefetch_tokens_occupied -= len(prefetch_key)
        self.prefetch_loaded_tokens_by_reqid[req_id] = loaded_from_storage
        logger.info(
            "HiCache prefetch %s req=%s completed_local=%d completed_synced=%d "
            "matched=%d loaded=%d released=%d occupied=%d",
            "dropped" if insert_result.host_insert_dropped else "success",
            req_id,
            completed_tokens,
            min_completed_tokens,
            insert_result.prefix_len,
            loaded_from_storage,
            released_tokens,
            self.cache_controller.prefetch_tokens_occupied,
        )
```

评论区精华

核心讨论如下：

- 方案选择：丢弃而非修补。hzh0425 在关联 PR #30106 的评论里指出："Perhaps it's better to simply drop the prefetch result rather than trying to patch it"，作者立刻采纳，因为丢弃更简单且避免额外 edge case。
- write-through 限定。hzh0425 在 review 中提出 "We need to check here: the device-parent requirement only applies in non-write-back modes"，即 backup 父节点约束只在 write-through 下成立，write-back 没有该要求，修复必须加 not self.is_write_back 门控，作者据此补充了对应测试。
- 测试组织。hzh0425 要求不新建文件："Don't create a new test; just add it directly to the existing test_unified_radx_cache_unit.py"，作者将测试并入既有文件。
- 可观测性。hzh0425 要求为 drop 事件加日志（"pls add a loggger for the drop event"），最终落地为 HiCache prefetch dropped ... info 日志。
- 测试有效性质疑。decajoin 发现 hzh0425 新增的 test_dropped_prefetch_releases_all_host_resources 存在 TypeError: 'Mock' object is not iterable，且因 setUpModule 跳过而未被 CI 捕获；hzh0425 请求协助修复，decajoin 修复了 mock 未设置 operation.pool_transfers 与缺少存储后端导致的 host_mem_release_queue 缺失问题。

- rebase 连带发现。decajoin 在 rebase 后指出 #29901 新增的 test_shallower_crossing_backs_up_above_backuped_middle 构造的正是本 PR 视为 corruption 的非法状态（backuped 节点挂在 unbacked 父节点下），在 clean main 上该状态已无法通过 sanity_check 且驱逐会留下脏的 host_leaves 条目，但测试从未调用 sanity_check 而未被发现；该文件被 #31812 跳过，CI 暂不受影响，但文件重新启用后该用例会失败，需要后续跟进。
- 舍弃修补方案，直接丢弃 refill (design): 统一采用丢弃方案：anchor 未备份时放弃本次 best-effort 预取，而不是反向备份父节点。
- drop 逻辑必须限定 write-through（非 write-back） (correctness): 修复加入 not self.is_write_back 门控，write-back 下保留 refill，并补充对应测试。
- 回归测试并入既有文件 (style): 测试合入 test_unified_radix_cache_unittest.py。
- drop 事件需要日志 (other): 在 insert_host 与 check_prefetch_progress 中均增加 info 日志，标明 dropped/success 与 released_tokens。
- 新增测试存在 Mock 未迭代错误且被整体跳过 (testing): 测试被修复并通过本地验证；但模块级跳过仍在，CI 守护待 #31812 解除后恢复。
- rebase 后发现 #29901 测试构造了同款非法状态 (correctness): 该文件已被 #31812 跳过，CI 暂不受影响；但重新启用后该用例会失败，作为未解决的连带问题留待后续跟进。

风险与影响

- 风险：风险点如下：
- 预取命中率下降（有意的权衡）：write-through 下 anchor 未备份时 refill 被丢弃，会退化为一次 cache miss；PR body 明确接受该代价（hot path 仅多一次 node.backuped 检查）。若生产环境 device-only anchor 出现频繁，可能放大外部存储读取压力，建议观察 prefetch_loaded_tokens 指标。
- 回归测试当前未生效：test_unified_radix_cache_unittest.py 被 pytestmark 与 setUpModule 整体跳过，新增的 TestUnifiedRadixPrefetchCorruption 并未在 CI 运行，守护作用打折；且模块重新启用后，test_shallower_crossing_backs_up_above_backuped_middle（#29901 引入）预期会因本修复而失败，属于未解决的连带问题。
- 契约字段漏处理风险：InsertResult.host_insert_dropped 作为新的数据契约字段，若有其他调用方未处理该标志，可能造成 host 页面泄漏或错误提交；当前仅 check_prefetch_progress 一处消费，需关注未来新增调用方。
- rebase 迁移正确性：修复逻辑从 _insert_helper_host 迁移到 UnifiedTreeCore.insert_host()（#29901 拆分），逻辑不变但需确认该内核被其他路径调用时不引入语义差异。
- 影响：影响评估：
- 用户 / 系统层面：修复了 HiCache（混合 KV 缓存）在 write-through 策略下的调度器崩溃问题，直接影响使用外部存储预取的长上下文场景（如 DeepSeek 系列、Mamba 混合模型），从“偶发进程级崩溃”变为“一次可预期的 cache miss”。
- 性能影响：正常路径零额外开销；丢弃路径多一次 host 页面释放与日志，热路径每次 prefetch 多一次成员检查，可忽略。

- 团队层面：合入者 hzh0425 深度参与方案定型与测试改进，说明 unified radix cache 是当前核心维护方向；本 PR 与 #30106 互补覆盖不同缓存类的同一类不变量问题。
- 风险标记：核心缓存路径变更，回归测试当前被跳过，预取命中率下降，连带测试遗留待处理

关联脉络

- PR #30106 同一根因、不同缓存类的修复：PR body 与 issue 评论均提及：#30106 解决相同的不变量破坏问题，但针对不同缓存类，文件无重叠，两者互补；hzh0425 的“丢弃而非修补”建议正来自该 PR 的 review。
- PR #29901 TreeCore 拆分将 _insert_helper_host 移至 UnifiedTreeCore.insert_host: 本 PR rebase 时修复逻辑随之迁移到 UnifiedTreeCore.insert_host；且 #29901 引入的 test_shallower_crossing_backs_up_above_backuped_middle 与本修复存在语义冲突，是未解决的连带问题。
- PR #31812 test_unified_radix_cache_unittest.py 模块级跳过：本 PR 的回归测试所在模块目前被 #31812 跳过，CI 未实际执行新用例；解除跳过时需同步处理 test_shallower_crossing 的失败风险。