PR #36382 完整报告

sgl-project/sglang

[HiCache] Key storage prefetch by the request namespace

合并时间：2026-08-29 02:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36382

执行摘要

- 一句话：HiCache 存储预取改按请求命名空间定键，修复双重释放
- 推荐动作：值得精读，尤其是两个设计决策：预取键的命名空间应从请求而不是 anchor 派生（anchor 无命名空间），以及用断言让不一致场景大声失败而不是静默降级。建议后续补充一个多租户 + 分层缓存 + L3 存储的端到端测试，验证修复后的预取命中与无 double-free；同时确认调度器 _prefetch_kvcache 路径上的断言异常不会中断调度循环。

功能与动机

PR body 指出：L3 存储预取键的命名空间（extra_key / cache_salt）来自 anchor 节点而非发起请求，根 anchor 完全没有命名空间，因此非默认命名空间的请求会在错误键下暂存预取 span，随后请求自身的 insert 重新拥有这些槽位，导致 span 被释放两次（double free）。触发条件为 extra_key 或 cache_salt 被使用（按租户隔离缓存、盐化前缀缓存）时，同时启用分层缓存与 L3 存储后端。这是多租户缓存隔离场景下才会暴露的正确性缺陷，且属于隐蔽的内存所有权错误，而非单纯的命中率问题。

实现拆解

1. 根因定位：unified_radix_cache.py 的 prefetch_from_storage 通过 self.tree_core.prefetch_anchor_info(last_host_node_id) 从 anchor 节点取命名空间；根 anchor 返回 None，而请求可能是非默认命名空间，两者不一致。
2. 核心修复：为 prefetch_from_storage 增加 extra_key / cache_salt 关键字参数，并用它们构造 RadixKey；anchor 的命名空间仍被读取并断言与请求一致——anchor 有命名空间且不匹配时直接抛异常，保证“大声失败”而非静默 mis-key。
3. 调用点同步：scheduler.py 的 _prefetch_kvcache 与 disaggregation/decode_hicache_mixin.py 的 _start_hicache_prefetch 分别传入 req.extra_key / req.cache_salt；hiradix_cache.py 增加同名参数保持调度器 duck-typed 调用兼容（缓存模式写穿 anchor 在请求自身路径上，命名空间由 last_host_node.key 携带，参数不参与键构造）。
4. buffer 模式防御：buffer_mode/pipeline.py 的 _StagedPrefetch 新增 cache_salt 字段，stage_completed_prefetch 记录预取键命名空间；init_load_back 消费 staged hold 时发现命名空间与请求不一致则直接丢弃（防御性分支，正常路径不可达），splice 重建 RadixKey 改用 staged 自身记录的 extra_key 与 cache_salt。

5. 测试配套：test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py 中 4 处 buffer-mode 消费辅助构造的 request mock 补齐 extra_key / cache_salt 属性。

关键文件：

- python/sglang/srt/mem_cache/buffer_mode/pipeline.py（模块 缓冲管线；类别 source；类型 core-logic；符号 _StagedPrefetch, stage_completed_prefetch, init_load_back）：修复的核心防御层：staged prefetch 记录完整命名空间，splice 时校验并重建 RadixKey，防止错误命名空间发布导致的 double-free。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 统一缓存；类别 source；类型 core-logic；符号 prefetch_from_storage）：根因修复点：预取键从 anchor 命名空间改为请求命名空间，并通过断言让不一致场景大声失败。
- python/sglang/srt/disaggregation/decode_hicache_mixin.py（模块 解耦调度；类别 source；类型 core-logic；符号 _start_hicache_prefetch）：decode 侧预取调用点需要转发请求命名空间，否则修复不生效。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _prefetch_kvcache）：前向流调度侧的预取调用点补传命名空间，是整个调用链的一环。
- python/sglang/srt/mem_cache/hiradix_cache.py（模块 分层缓存；类别 source；类型 core-logic；符号 prefetch_from_storage）：经典分层缓存实现需同步签名，避免 duck-typed 调用抛 TypeError。
- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 单元测试；类别 test；类型 test-coverage；符号 _consume_staged_prefetch）：保证 buffer-mode 消费辅助函数在新字段下可用，覆盖回归。

关键符号：prefetch_from_storage (UnifiedRadixCache), prefetch_from_storage (HiRadixCache), _start_hicache_prefetch, _prefetch_kvcache, stage_completed_prefetch, init_load_back

关键源码片段

python/sglang/srt/mem_cache/buffer_mode/pipeline.py

修复的核心防御层：staged prefetch 记录完整命名空间，splice 时校验并重建 RadixKey，防止错误命名空间发布导致的 double-free。

buffer_mode/pipeline.py：staged prefetch 的命名空间守卫与键重建。

```
class _StagedPrefetch(msgspec.Struct):
    """A completed buffer-mode fetch parked until prefill admission: only
    the op-owned host bounce exists (no device state, nothing in the tree).
    """

    req_id: str
    key_tokens: list[int]
    extra_key: Optional[str]
    # cache_salt 与 extra_key 共同构成命名空间：splice 重建 RadixKey 时，
    # 必须用 staged 自身的命名空间，而不是从 anchor 节点推导。
```

```
    cache_salt: Optional[str]
    matched_len: int
    num_tokens: int
    occupied_tokens: int
    host_indices: torch.Tensor
    aux_xfers: list[PoolTransfer]
    hash_values: list[str]
    operation_id: int

# 位于 init_load_back 消费 staged prefetch 的核心分支：
# 命名空间不匹配的 hold 必须在 splice 之前丢弃——在错误命名空间下发布
# span 会导致槽位重复归属与双重释放。该守卫是防御性兜底，正常路径下
# 预取键已由请求命名空间派生，此分支不可达。
if f.extra_key != req.extra_key or f.cache_salt != req.cache_salt:
    logger.error(
        "HiCache staged prefetch dropped req=%s reason=namespace "
        "staged=%s req=%s",
        req.rid,
        (f.extra_key, f.cache_salt),
        (req.extra_key, req.cache_salt),
    )
    return _drop()

# 重建 RadixKey 时使用 staged 记录的 extra_key 与 cache_salt，
# 与请求插入路径的键保持一致，避免命中错误命名空间的存储数据。
splice_key = RadixKey(
    array("q", f.key_tokens),
    extra_key=f.extra_key,
    is_bigram=cache.tree_core.is_eagle,
    cache_salt=f.cache_salt,
).page_aligned(cache.page_size)
```

python/sglang/srt/mem_cache/unified_radix_cache.py

根因修复点：预取键从 anchor 命名空间改为请求命名空间，并通过断言让不一致场景大声失败。

unified_radix_cache.py：按请求命名空间构造预取键与一致性断言。

```
def prefetch_from_storage(
    self,
    req_id: str,
    last_host_node_id: NodeId,
    new_input_tokens: list[int],
    last_hash: Optional[str] = None,
    prefix_keys: Optional[list[str]] = None,
    matched_prefix_tokens: Optional[list[int]] = None,
    extra_key: Optional[str] = None,
    cache_salt: Optional[str] = None,
) -> None:
```

```
if not self.enable_storage or self.cache_controller is None:
    return

buffer_mode = self.host_memory_mode == "buffer_only"
# 预取 span 用请求自身的命名空间定键，而不是 anchor 节点的命名空间：
# 根 anchor 没有命名空间，沿用 anchor 会把非默认命名空间请求的 span
# 挂到错误键下，随后被请求自己的 insert 重新拥有（双重释放）。
anchor_extra_key, anchor_cache_salt = self.tree_core.prefetch_anchor_info(
    last_host_node_id
)
# anchor 有命名空间时必须与请求一致，否则断言失败并大声报错，
# 避免静默 mis-key 掩盖真正的配置错误。
assert (anchor_extra_key is None or anchor_extra_key == extra_key) and (
    anchor_cache_salt is None or anchor_cache_salt == cache_salt
), (
    f"prefetch anchor namespace {(anchor_extra_key, anchor_cache_salt)} "
    f"!= request namespace {(extra_key, cache_salt)}"
)
prefetch_key = RadixKey(
    new_input_tokens,
    extra_key=extra_key,
    is_bigram=self.tree_core.is_eagle,
    cache_salt=cache_salt,
).page_aligned(self.page_size)
prefetch_length = len(prefetch_key)
# 后续逻辑不变：根据 prefetch_length 决定是否执行存储预取。
```

评论区精华

PR 无实质 review 讨论，唯一评论是作者在关联 issue 中发布的 /tag-and-rerun-ci 命令。设计权衡主要来自 PR body 自述：anchor 有命名空间时读取并断言其与请求一致，使真 mismatch 大声失败而非静默 mis-key；buffer 模式的命名空间守卫被明确定位为 defence-in-depth，在预取键由请求派生后不可达；hiradix_cache 增加两个未参与定键的参数，是为了保持调度器 duck-typed 调用兼容。若线上断言触发，说明 anchor 命名空间与请求不一致，需要检查分层缓存节点的跨租户复用。

- PR 无实质 review 讨论，仅作者触发 CI 重跑 (other)：无未决疑虑；PR 由作者自行合并，设计取舍见 PR body。

风险与影响

- 风险：
  - unified_radix_cache.py 新增的断言是硬失败路径：若 anchor 节点因节点复用携带与请求不同的命名空间，prefetch 会抛 AssertionError。decode 侧有 try/except 会降级为 L2-only 恢复；调度器 _prefetch_kvcache 路径未见显式捕获，需确认异常是否会传播到调度循环。

    - 命名空间比较是严格相等（None 与 "" 不等价）：若调用方以空字符串表示默认命名空间而请求使用 None，可能触发防御性 drop 或断言失败。
    - prefetch_from_storage 是调度器 duck-typed 调用接口：除 hiradix_cache.py 外，第三方 / 自定义 tree_cache 实现若未同步增加参数，调用将抛 TypeError。
    - 新守卫依赖"预取键由请求派生"这一不变式；若未来某路径绕过该不变式直接构造 staged prefetch，将依赖 logger.error 与 drop 兜底，可能掩盖真实 bug。
- 影响：
    - 行为变化仅涉及启用 extra_key / cache_salt（多租户隔离、盐化前缀缓存）且开启分层缓存与 L3 存储的场景；默认无命名空间场景行为不变。
    - 修复消除错误命名空间发布导致的 double-free，存储预取命中数据也更准确；热路径无额外开销，断言读取的是已获取状态。
    - 对团队而言，这是 HiCache 正确性系列修复之一，为多租户缓存隔离场景提供更稳的发布前提。
    - 风险标记：核心缓存路径变更，新增断言可能触发线上异常，duck-typed 接口兼容风险，缺少命名空间端到端测试

关联脉络

- PR #36227 [HiCache] Retry L3 storage prefetch after a missed attempt: 同一 L3 存储预取机制，改动文件高度重叠（scheduler.py、unified_radix_cache.py、buffer_mode/pipeline.py、hiradix_cache.py），本 PR 修复其预取键命名空间归属问题。
- PR #36738 [HiCache] Fence load-back behind the forward stream: 同涉 buffer_mode/pipeline.py 与调度并发正确性，与 double-free/ 所有权主题相邻。
- PR #35931 [HiCache] Reject load-back specs that claim nodes pinned by an in-flight load-back: 同属 HiCache load-back 所有权 / 双重释放正确性修复，改动 unified_radix_cache.py。
- PR #36425 HiCache: avoid unnecessary all-reduce in check_prefetch_progress: 同属 unified_radix_cache.py 存储预取链路优化，演进线路相关。