PR #23678 完整报告

sgl-project/sglang

feat: Add KV events for Mamba radix cache

合并时间：2026-05-09 02:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23678

执行摘要

- 一句话：为 Mamba radix cache 添加 KV 事件发射
- 推荐动作：值得精读。PR 展示了如何通过 Mixin 模式在不破坏继承层次的情况下为多缓存类添加正交功能。split_node_hash_value 设计、Mamba-only 驱逐静默策略值得同类系统参考。

功能与动机

PR body 指出：MambaRadixCache currently inherits BasePrefixCache.take_events(), so enabling --kv-events-config does not publish KV cache placement events for the Mamba radix path. Downstream KV-aware routers then lose precise cache placement information and fall back to approximate routing.

实现拆解

1. 提取事件发射基础设施：创建 python/sglang/srt/mem_cache/events.py，定义 KVCacheEventMixin，包含 _record_store_event、_record_remove_event、_record_all_cleared_event 和 take_events。同时将原内嵌在 RadixCache 中的 compute_node_hash_values 和 split_node_hash_value 迁移到 mem_cache/utils.py。
2. 改造 RadixCache：将其基类改为 KVCacheEventMixin, BasePrefixCache，删除重复事件方法。
3. 接入 MambaRadixCache：添加 kv_event_queue，在 _insert_helper、_evict_leaf_node、_split_node 等位置调用事件发射；Mamba-only tombstone 清除保持静默。
4. 接入 HiMambaRadixCache：在 load_back、writing_check、_evict_to_host、_evict_regular、_evict_host_leaf、_delete_tombstone_leaf 中插入事件，区分 StorageMedium.GPU/CPU。
5. 测试：新增单元测试 test_mamba_radix_cache_kv_events 和 test_mamba_radix_cache_kv_events_split_hash；集成测试 test_qwen35_hicache.py 验证端到端事件流。

关键文件：

- python/sglang/srt/mem_cache/events.py（模块 缓存层；类别 source；类型 core-logic；符号 KVCacheEventMixin, _record_store_event, _record_remove_event, _record_all_cleared_event）：核心新增文件，定义 KVCacheEventMixin，所有事件发射

逻辑汇聚于此。

- python/sglang/srt/mem_cache/radix_cache.py（模块 缓存层；类别 source；类型 refactor；符号 RadixCache, compute_node_hash_values, split_node_hash_value）：重构 RadixCache 继承 KVCacheEventMixin，删除重复事件方法，简化维护。
- python/sglang/srt/mem_cache/utils.py（模块 工具层；类别 source；类型 core-logic；符号 compute_node_hash_values, split_node_hash_value）：新增 compute_node_hash_values 和 split_node_hash_value 工具函数，供所有缓存类共享。
- python/sglang/srt/mem_cache/mamba_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 MambaRadixCache, split_node_hash_value）：MambaRadixCache 接入事件系统，在插入、驱逐、分裂时触发事件。
- python/sglang/srt/mem_cache/hi_mamba_radix_cache.py（模块 缓存层；类别 source；类型 dependency-wiring）：HiMambaRadixCache 增加事件发射，区分 GPU/CPU 介质，确保层次化缓存场景下事件正确。
- test/registered/unit/mem_cache/test_mamba_unittest.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_mamba_radix_cache_kv_events, test_mamba_radix_cache_kv_events_split_hash, _setup_tree_and_allocator）：单元测试验证事件发射基本路径和分裂哈希稳定性。

关键符号：KVCacheEventMixin._record_store_event, KVCacheEventMixin._record_remove_event, KVCacheEventMixin._record_all_cleared_event, KVCacheEventMixin.take_events, compute_node_hash_values, split_node_hash_value

关键源码片段

python/sglang/srt/mem_cache/events.py

核心新增文件，定义 KVCacheEventMixin，所有事件发射逻辑汇聚于此。

```
# events.py — KVCacheEventMixin
from typing import Any
from sglang.srt.disaggregation.kv_events import (
    AllBlocksCleared, BlockRemoved, BlockStored, StorageMedium,
)
from sglang.srt.mem_cache.utils import compute_node_hash_values, hash_str_to_int64

class KVCacheEventMixin:
    # KV cache placement event emission mixin.

    def _record_store_event(self, node: Any, medium=None):
        # 仅在启用配置时执行
        if not self.enable_kv_cache_events:
            return
        if medium is None:
            medium = StorageMedium.GPU
        # 延迟计算节点哈希值
        if node.hash_value is None:
            node.hash_value = compute_node_hash_values(node, self.page_size)
```

```
        # 获取父节点最后一个块哈希用于链式连接
        parent_block_hash = None
        if node.parent is not None and node.parent != self.root_node:
            if node.parent.hash_value and len(node.parent.hash_value) > 0:
                parent_block_hash = hash_str_to_int64(node.parent.hash_value[-1])
        page_index = 0
        logical_len = len(node.key)
        is_bigram = node.key.is_bigram
        raw = node.key.token_ids
        for start in range(0, logical_len, self.page_size):
            end = min(start + self.page_size, logical_len)
            if end <= start:
                continue
            # 对于 bigram 模式，token_ids 取连续二元组
            if is_bigram:
                page_tokens = [(raw[j], raw[j+1]) for j in range(start, end)]
            else:
                page_tokens = raw[start:end]
            block_hash = hash_str_to_int64(node.hash_value[page_index])
            self.kv_event_queue.append(
                BlockStored(
                    block_hashes=[block_hash],
                    parent_block_hash=parent_block_hash,
                    token_ids=page_tokens,
                    block_size=len(page_tokens),
                    lora_id=None,
                    medium=medium,
                )
            )
            parent_block_hash = block_hash
            page_index += 1
```

python/sglang/srt/mem_cache/utils.py

新增 compute_node_hash_values 和 split_node_hash_value 工具函数，供所有缓存类共享。

```
# utils.py — compute_node_hash_values 与 split_node_hash_value
def compute_node_hash_values(node, page_size):
    # 计算节点的逐块 SHA256 哈希值，用于 KV 事件块标识
    hash_values = []
    parent_hash = None
    if node.parent is not None and node.parent.hash_value is not None:
        if len(node.parent.key) > 0 and len(node.parent.hash_value) > 0:
            parent_hash = node.parent.hash_value[-1]
    logical_len = len(node.key)
    for start in range(0, logical_len, page_size):
        end = min(start + page_size, logical_len)
        if end <= start:
            continue
        hash_val = node.key.hash_page(start, end, parent_hash)
```

```
            hash_values.append(hash_val)
            parent_hash = hash_val
        return hash_values

    def split_node_hash_value(child_hash_value, split_len, page_size):
        # 拆分 hash_value 列表，用于节点分裂时保持哈希链一致
        if child_hash_value is None:
            return None, None
        if page_size == 1:
            split_pages = split_len
        else:
            split_pages = split_len // page_size
        new_node_hash = child_hash_value[:split_pages]
        child_hash = child_hash_value[split_pages:]
        return new_node_hash, child_hash

    # 注：该函数被 RadixCache、MambaRadixCache、HiMambaRadixCache 的 _split_node 共享
```

评论区精华

- 事件方法抽象：hzh0425 注意到 MambaRadixCache 与 RadixCache 中 _record_store_event 等几乎相同，建议抽取复用。作者将其移至 events.py 的 KVCacheEventMixin 中。
- split 兼容性：hzh0425 质疑在 MambaRadixCache._split_node 添加哈希拆分是否干扰 HiMambaRadixCache。作者论证 MambaRadixCache 必须同步切片哈希值，且 Hi 路径通过 super() 自动继承，不冲突。
- 封装性担忧：hzh0425 对 compute_node_hash_values 在外部设置 node.hash_value 表示疑虑。作者指出该 lazy compute 模式已在 RadixCache 中存在，PR 仅迁移位置；hzh0425 最终理解。
- 集成测试要求：hzh0425 要求增加 server 级别测试，作者在 test_qwen35_hicache.py 中补充。
- 事件方法抽象复用 (design): 作者将相关方法移至 events.py 的 KVCacheEventMixin 中，实现统一复用。
- split_node_hash_value 与 HiMambaRadixCache 兼容性 (correctness): 作者论证 MambaRadixCache 现已跟踪 hash_value，拆分时必须同步切片，且 Hi 路径通过 super()._split_node() 自动继承，不冲突。
- 外部函数修改节点 hash 值的封装性 (design): 作者指出此 lazy compute 模式在现有 RadixCache 中已存在，PR 仅迁移位置；hzh0425 最终表示理解。
- 添加集成测试 (testing): 作者在 test/registered/4-gpu-models/test_qwen35_hicache.py 中新增集成测试。

风险与影响

- 风险：

  - 事件队列内存：下游消费速度可能跟不上，队列可能增长。但原始 RadixCache 已有相同问题，且仅在启用配置时激活，风险低。
- splithash同步性：split_node_hash_value假设page_size固定，若动态变化则索引错位。当前 page_size 常量，无风险。
- Mamba-only silence：设计选择，不影响全注意力 KV 事件。如未来需要追踪 Mamba 状态，需扩展 schema。
- 兼容性：HiMambaRadixCache 新增 hash_value 字段，不影响读写，已在测试中验证。
- 影响：
  - 用户影响：启用了 --kv-events-config 的 Mamba 推理现在能正确发射 KV 事件，下游 Dynamo 等路由器可精确路由，避免退化为近似路由。
- 系统影响：CPU 增加哈希计算和队列操作开销，仅在启用配置时产生。
- 团队影响：KVCacheEventMixin 使新缓存类可零成本复用事件逻辑，降低维护成本。
- 风险标记：事件队列内存增长，split hash 同步性，lazy compute 封装性

关联脉络

- 暂无明显关联 PR