PR #23631 完整报告

sgl-project/sglang

[HiCache][SPEC] fix: normalize storage prefetch key

合并时间：2026-04-29 06:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23631

执行摘要

- 一句话：修复 HiCache EAGLE 预取键不归一化的 bug
- 推荐动作：值得精读，展示了如何通过统一 key 类型（RadixKey）解决数据视图不一致导致的 bug。设计决策（bigram 视图通过 RadixKey 构造而非手动转换）值得推广。建议后续补充单元测试。

功能与动机

HiCache 在 EAGLE 场景下，存储预取路径仍使用旧的手动 bigram 键（通过 convert_to_bigram_key 转换），而加载回后以非 bigram 的 RadixKey 值插入，导致 radix 树匹配 / 分裂无法进行或挂起。PR body 明确指出了根因。

实现拆解

1. 移除旧的 bigram 转换依赖：在 hiradix_cache.py 中删除 from sglang.srt.mem_cache.utils import convert_to_bigram_key 导入。
2. 统一构造 RadixKey 作为预取键：在 prefetch_from_storage 方法中，用 RadixKey(new_input_tokens, extra_key=..., is_bigram=self.is_eagle) 替代 convert_to_bigram_key，并通过 page_aligned 方法对齐页面。
3. 重命名变量并简化插入逻辑：在 check_prefetch_progress 方法中，将原来名为 token_ids 的变量（实际存的是预取键）重命名为 prefetch_key，直接使用 _insert_helper_host 传入该键，不再手动构造新的 RadixKey。
4. 关联配套调整：同步更新 prefetch_tokens_occupied 的统计和 ongoing_prefetch 的存储键类型。
5. 无测试文件变更：本 PR 仅修改了核心源码，未新增或修改测试。

关键文件：

- python/sglang/srt/mem_cache/hiradix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 check_prefetch_progress, prefetch_from_storage）：核心变更文件，所有修改集中在 check_prefetch_progress 和 prefetch_from_storage 两个方法，包括移除旧 bigram 转换、统一使用 RadixKey、变量重命名等。

关键符号：prefetch_from_storage, check_prefetch_progress, _insert_helper_host

关键源码片段

python/sglang/srt/mem_cache/hiradix_cache.py

核心变更文件，所有修改集中在 check_prefetch_progress 和 prefetch_from_storage 两个方法，包括移除旧 bigram 转换、统一使用 RadixKey、变量重命名等。

```
def prefetch_from_storage(
    self,
    req_id: str,
    new_input_tokens: List[int],
    last_host_node: TreeNode,
    last_hash: Optional[str] = None,
    prefix_keys: Optional[List[str]] = None,
):
    # 统一通过 RadixKey 构造预取键，EAGLE 时自动使用 bigram 视图
    prefetch_key = RadixKey(
        new_input_tokens,
        extra_key=last_host_node.key.extra_key,
        is_bigram=self.is_eagle,
    )
    # 按 page 对齐，确保预取长度是 page_size 的整数倍
    prefetch_key = prefetch_key.page_aligned(self.page_size)
    prefetch_length = len(prefetch_key)

    if (
        not self.enable_storage
        or prefetch_length < self.prefetch_threshold
    ):
        return

    # ... (allocator 相关的 pool 检查略去 ) ...

    operation = self.cache_controller.prefetch(
        req_id, host_indices, prefetch_key, last_hash, prefix_keys,
        **self._get_extra_pools(),
    )
    self.ongoing_prefetch[req_id] = (
        last_host_node,
        prefetch_key, # 注意：这里存的是 RadixKey，不是 raw token list
        host_indices,
        operation,
    )
    self.cache_controller.prefetch_tokens_occupied += len(prefetch_key)

def check_prefetch_progress(self, req_id: str) -> bool:
    # ... 异常处理略 ...
    # 从 ongoing_prefetch 中取出预取键（已重命名为 prefetch_key）
    last_host_node, prefetch_key, host_indices, operation = self.ongoing_prefetch[
        req_id
    ]
    # ... 终止条件判断略 ...
```

```
min_completed_tokens = completed_tokens_tensor.item()
# 直接截取 RadixKey 对象的切片，行为由 __getitem__ 定义
fetched_key = prefetch_key[:min_completed_tokens]
written_indices = host_indices[:min_completed_tokens]
matched_length = self._insert_helper_host(
    last_host_node,
    fetched_key, # 直接传入 RadixKey，不再手动构造
    written_indices,
    hash_value[: min_completed_tokens // self.page_size],
)
# ... 释放资源略 ...
```

评论区精华

review 中 gemini-code-assist[bot] 提出变量命名建议：token_ids[:min_completed_tokens] 中的 token_ids 实际是 RadixKey 对象，切片操作调用的是 RadixKey.__getitem__，容易误导。该建议被作者采纳，在后续提交中将变量重命名为 prefetch_key。

- 变量命名：token_ids 应改为 prefetch_key 以避免歧义 (style): 作者采纳建议，在后续 commitic 中将变量重命名为 prefetch_key。

风险与影响

- 风险：主要风险在于 RadixKey 的 __getitem__ 行为与旧 token list 切片是否完全对齐，特别是 bigram 模式下序列长度减半的处理。如果 RadixKey.__getitem__ 实现有边界条件差异，可能导致 min_completed_tokens 切片异常，进而影响预取完成后 host cache 插入的正确性。该 PR 仅通过静态编译和 e2e 测试验证，缺少单元测试覆盖。建议在未来的 PR 中补充 RadixKey 切片在 bigram 模式下的单元测试。
- 影响：直接影响 HiCache 存储系统的正确性，修复后 EAGLE 投机解码与 HiCache 预取可正常协作，避免 radix 树死锁；对非 EAGLE 场景（is_bigram=False）无功能变化。纯源码变更，无外部接口改动。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #23387 [HiCache][SPEC] fix: empty key after page alignment in match_prefix: 同为 HiCache + SPEC 相关的 bugfix，修改同一个文件 hiradix_cache.py，解决 radix 匹配相关问题。