PR #42656 完整报告

vllm-project/vllm

Apply LRU policy only to proper cache entries

合并时间：2026-06-17 05:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42656

执行摘要

- 一句话：优化 KV cache 释放策略，仅对缓存条目应用 LRU
- 推荐动作：值得精读，特别是 free_blocks 的设计决策——将分类逻辑下沉到基础设施，对调用方透明。是经典的缓存替换策略改进案例。

功能与动机

PR body 指出：partial blocks 没有 hash，不会被缓存复用，但原实现中它们与正常缓存 block 一起进入 LRU 队列，导致它们过早驱逐真正的缓存条目。例如在 memory tight 场景下两个 prompt 的 partial blocks 互相驱逐对方缓存，或在 memory abundant 场景下大量并发 prompt 的 partial blocks 完全冲掉 KV cache，造成本应命中的缓存 miss。

实现拆解

1. 重构 free_blocks 方法（vllm/v1/core/block_pool.py）：移除 prepend 参数，内部遍历输入 block，根据 block_hash is None 分组，无 hash block 通过 prepend_n 插入头部，有 hash block 通过 append_n 插入尾部。
2. 简化 remove_skipped_blocks（vllm/v1/core/single_type_kv_cache_manager.py）：原分别收集 cached/uncached block 并两次调用 free_blocks（带 prepend），现统一收集后单次调用 free_blocks，分类下沉至 block pool。
3. 移除 SlidingWindowManager.free 的重复逻辑：原 sliding window 管理器有独立 free 方法做同样分类，现删除，通过传递 reversed(req_blocks) 保持顺序，复用统一 free_blocks。
4. 测试适配（tests/v1/core/test_prefix_caching.py）：更新 free queue 顺序断言，反映无 hash block 出现在队列头部。

关键文件：

- vllm/v1/core/block_pool.py（模块 缓存层；类别 source；类型 core-logic；符号 free_blocks）：核心修改文件：free_blocks 方法移除 prepend 参数，内部根据 block_hash 分类处理，分别 prepend 无 hash block 和 append 有 hash block，简化了接口并统一 behavior。
- vllm/v1/core/single_type_kv_cache_manager.py（模块 缓存层；类别 source；类型 core-logic；符号 free, remove_skipped_blocks）：简化 remove_skipped_blocks 并移除 SlidingWindowManager.free 中重复分类逻辑，统一使用新的 free_blocks。

- tests/v1/core/test_prefix_caching.py（模块 测试；类别 test；类型 test-coverage）：更新测试断言以反映新的 free queue 顺序，包括无 hash block 出现在头部，以及重用部分逻辑。

关键符号：free_blocks, remove_skipped_blocks, free

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

简化 remove_skipped_blocks 并移除 SlidingWindowManager.free 中重复分类逻辑，统一使用新的 free_blocks。

```
def remove_skipped_blocks(self, request_id: str, total_computed_tokens: int) -> None:
    # ... 计算 num_skipped_blocks 后，统一收集所有待释放块:
    removed_blocks: list[KVCacheBlock] = []
    for i in range(num_skipped_blocks - 1, -1, -1):
        if blocks[i] == self._null_block:
            break
        removed_blocks.append(blocks[i])
        blocks[i] = self._null_block
    # 单次调用 free_blocks，内部自动分类
    self.block_pool.free_blocks(removed_blocks)
```

评论区精华

njhill 建议简化并与 PR #43447 合并：去掉 prepend 参数，SlidingWindowManager.free 只需传 reversed(req_blocks) 而非重复分类逻辑。作者采纳并完成修改。

- 简化 free_blocks 接口并移除重复逻辑 (design): 作者 s3woz 采纳建议，移除了 prepend 参数和重复的 free 方法，并通过测试验证。

风险与影响

- 风险：free_blocks 接口移除 prepend 参数，所有调用方已同步修改，但若未来有外部扩展依赖该参数会编译失败。分类逻辑假设 block_hash is None 一定表示 scratch block，需确保 evict 后的 block 不会进入此路径（当前不会）。整体代码量减少，风险可控。
- 影响：正面影响所有使用 prefix caching 的模型，尤其是高并发、长 prompt、有限 GPU 内存场景。用户无需修改配置即可体验缓存命中率提升和延迟降低。内部接口简化，维护成本下降。
- 风险标记：核心路径变更，接口简化（移除参数），测试覆盖充分

关联脉络

- PR #43447 [Core] Optimize sliding window cache eviction: 本 PR 泛化了 #43447 的滑动窗口缓存释放优化，将分类逻辑统一到 block pool 中，覆盖更多场景。