PR #45939 完整报告

vllm-project/vllm

[1/N][Core] add partial prefix cache primitives

合并时间：2026-06-22 14:43

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45939

执行摘要

- 一句话：添加细粒度前缀缓存原语，支持混合模型部分匹配
- 推荐动作：推荐精读此 PR。它是典型的分步设计范例：先构建核心原语，再逐步对接上层。特别值得关注 cache_partial_block 的生命周期管理、BlockHashToBlockMap 的多值处理，以及前缀链式哈希的复用策略。

功能与动机

Hybrid full-attention + Mamba models require the full-attention cache block size to align with the Mamba state block size; with block-size granular prefix matching, effective cache-hit granularity is much coarser than what full attention could otherwise support (PR body). This PR is the intentionally small first step toward finer-grained prefix cache hits.

实现拆解

1. KVCacheBlock 扩展：在 vllm/v1/core/kv_cache_utils.py 中新增 _block_hash_num_tokens 字段及属性、set_block_hash 方法，以记录 hash 覆盖的 token 数，区分部分与全块注册。
2. BlockHashToBlockMap 增强：在 vllm/v1/core/block_pool.py 的 BlockHashToBlockMap 中添加 contain 方法，支持在多值映射时检查指定 block_id 是否存在，替代不稳健的 get_one_block。
3. 反向映射字典：在 BlockPool.__init__ 中增加 cached_block_hashes_by_block: dict[int, set[BlockHashWithGroupId]]，用于快速删除某个 block 注册的所有 hash key，保证清理完整。
4. 核心方法 cache_partial_block：实现部分前缀注册逻辑，计算边界 hash、处理替换（partial -> partial 升级）和部分转全块时的清理，并发送 BlockStored / BlockRemoved 事件。
5. 配套清理与事件机制：提取 _remove_cached_block_hashes、_emit_block_removed_events 等辅助方法，确保 eviction/reset 时级联删除所有部分 entry。
6. cache_full_blocks 适配：当新全块已存在部分 hash 时，先移除旧部分条目再注册全块 hash，完成 partial->full 升级。

7. 测试覆盖：新增 tests/v1/core/prefix_cache/test_partial_prefix_cache_primitives.py（460 行），覆盖边界哈希复用、KV 事件、替换、缓存命中等场景；存量测试适配新 API。

关键文件：

- vllm/v1/core/block_pool.py（模块 块池；类别 source；类型 core-logic；符号 contain, cache_partial_block, _get_partial_block_hash, _get_partial_block_parent_hash_and_start）：变更最核心的文件，新增 contain、cache_partial_block、_remove_cached_block_hashes 等方法，以及反向映射字典，实现部分前缀注册与清理。
- tests/v1/core/prefix_cache/test_partial_prefix_cache_primitives.py（模块 前缀缓存测试；类别 test；类型 test-coverage；符号 _auto_init_hash_fn, make_request, boundary_hash, cache_full_block_and_partial_tail）：全新的测试文件，全面覆盖边界哈希、KV 事件、替换与缓存命中，验证核心逻辑的正确性。
- vllm/v1/core/kv_cache_utils.py（模块 KV 缓存工具；类别 source；类型 core-logic；符号 block_hash_num_tokens, set_block_hash）：修改了 KVCacheBlock 的数据结构，新增 block_hash_num_tokens 与 set_block_hash，为部分注册提供存储支持。
- tests/v1/core/test_kv_cache_utils.py（模块 KV 工具测试；类别 test；类型 test-coverage）：存量测试适配新 API，将 block.block_hash = block_hash 替换为 block.set_block_hash(block_hash)。
- tests/v1/core/test_prefix_caching.py（模块 前缀缓存测试；类别 test；类型 test-coverage）：存量测试适配新 API，将 block.block_hash = block_hash 替换为 block.set_block_hash(block_hash)。

关键符号：BlockHashToBlockMap.contain, BlockPool.cache_partial_block, BlockPool._get_partial_block_hash, BlockPool._get_partial_block_parent_hash_and_start, BlockPool._remove_cached_block_hashes, BlockPool._emit_block_removed_events, BlockPool._insert_block_hash, KVCacheBlock.set_block_hash, KVCacheBlock.block_hash_num_tokens

关键源码片段

vllm/v1/core/block_pool.py

变更最核心的文件，新增 contain、cache_partial_block、_remove_cached_block_hashes 等方法，以及反向映射字典，实现部分前缀注册与清理。

```
class BlockHashToBlockMap:
    """Dictionary-like container for mapping block hash keys to cache blocks."""

    def contain(self, key: BlockHashWithGroupId, block_id: int) -> bool:
        """检查 key 是否映射到指定的 block_id。"""
        blocks = self._cache.get(key)
        if blocks is None:
            return False
        # 当多个 block 共用一个 hash 时，blocks 是 dict；否则是单个 KVCacheBlock
        if isinstance(blocks, KVCacheBlock):
```

```
            return blocks.block_id == block_id
        if isinstance(blocks, dict):
            return block_id in blocks
        self._unexpected_blocks_type(blocks)
        return False

    # 其他方法（get_one_block, insert, pop）保持不变

class BlockPool:
    def __init__(self, ...):
        # ... 已有初始化 ...
        self.cached_block_hash_to_block: BlockHashToBlockMap = BlockHashToBlockMap()
        # 新增加的反向映射：由 block_id 到其注册的所有 hash key 集合
        self.cached_block_hashes_by_block: dict[int, set[BlockHashWithGroupId]] = {}
        # ...

    def cache_partial_block(
        self,
        request: Request,
        block: KVCacheBlock,
        num_tokens: int,
        kv_cache_group_id: int,
        block_size: int,
    ) -> BlockHashWithGroupId | None:
        """为一个已存在的 block 注册部分前缀 hash。"""
        assert num_tokens % self.hash_block_size == 0, (
            f"num_tokens {num_tokens} 必须能被 hash_block_size {self.hash_block_size} 整除"
        )
        # 计算边界 hash（复用 prefix-chain 中的已有值）
        parent_hash, start_idx = self._get_partial_block_parent_hash_and_start(
            request, block, num_tokens, block_size
        )
        if parent_hash is None:
            return None
        partial_hash = self._get_partial_block_hash(
            request, parent_hash, start_idx, num_tokens, kv_cache_group_id
        )
        if partial_hash is None:
            return None
        # 如果该 block 已有 hash（部分或全块），先清理旧的
        if block.block_hash is not None:
            removed_hashes = self._remove_cached_block_hashes(block)
            self._emit_block_removed_events(removed_hashes)
            block.reset_hash()
        # 插入新的部分条目
        self._insert_block_hash(partial_hash, block, num_tokens=num_tokens)
        return partial_hash
```

vllm/v1/core/kv_cache_utils.py

修改了 KVCacheBlock 的数据结构，新增 block_hash_num_tokens 与 set_block_hash，为部分注册提供存储支持。

```
@dataclass(slots=True)
class KVCacheBlock:
    """KV-cache block metadata."""

    block_id: int
    ref_cnt: int = 0
    # 原有 block_hash 字段
    _block_hash: BlockHashWithGroupId | None = None
    # 新增：被 _block_hash 覆盖的 prefix token 数量，部分注册时小于 block_size
    _block_hash_num_tokens: int | None = None

    @property
    def block_hash_num_tokens(self) -> int | None:
        return self._block_hash_num_tokens

    def set_block_hash(
        self,
        block_hash: BlockHashWithGroupId,
        num_tokens: int | None = None,
    ) -> None:
        # 强制 block 尚未设置任何 hash，防止覆盖
        assert self.block_hash is None and self._block_hash_num_tokens is None, (
            "The block already has a hash. This should not happen."
        )
        self._block_hash = block_hash
        self._block_hash_num_tokens = num_tokens

    def reset_hash(self):
        """Reset the block hash when the block is evicted."""
        self._block_hash = None
        self._block_hash_num_tokens = None
```

评论区精华

Reviewer ivanium 提出多项关键意见：

- 将 cache_block_alias 重命名为 cache_partial_block；已被采纳。
- 建议新增 contain 方法；已实现。
- 要求 set_block_hash 同时断言 block_hash_num_tokens 为 None；已采用。
- 删除原有的 block_hash setter，强制使用 set_block_hash；已执行。
- 对于 cache_partial_block，应确保 num_tokens 对齐 hash_block_size；以 assert 形式加入。
- 未解决的关注点：调用方（如 SimpleCPUOffloader）可能误将 block_hash != None 视为全块缓存；应在后续 PR 中审计。

- 函数命名：cache_block_alias -> cache_partial_block (design): 作者采纳，最终使用 cache_partial_block。
- BlockHashToBlockMap 缺乏检查指定 block_id 的能力 (correctness): 作者新增 contain 方法，直接检查 key 是否映射到给定 block_id。
- KVCacheBlock 状态一致性与调用方审计 (correctness): 已记录为 TODOs，作为后续 PR 的已知风险；核心逻辑通过断言保护。
- BlockHashListWithBlockSize 缓存失效 (performance): 当前 PR 添加了 block_hashes.clear_cached_views() 调用，但需确认所有 token 更新路径均能触发失效。

风险与影响

- 风险：
  1. 调用方语义假设风险：SimpleCPUOffloader 等代码可能将 block_hash != None 解释为全块缓存，但此 PR 后 block_hash 可能只代表部分边界。需要逐点审计。
  2. 反向映射 cached_block_hashes_by_block 在大量 block（>100K）下会存储过多 set 对象，可能增加内存与 GC 压力。
  3. 部分 entry 的替换 / 升级逻辑在并发或异常路径下可能导致 hash 指向错误或已释放的 block。
  4. BlockHashListWithBlockSize 在 append_output_token_ids 后清除了缓存，但流式续写可能绕过该清除，导致使用陈旧视图。 - 影响：当前 PR 未暴露 CLI 参数，不影响直接用户体验；但修改了 V1 引擎核心块池和前缀缓存的基础数据结构。所有依赖 prefix cache 的模块（调度器、offloading、vLLM 原生模型）均受隐含影响，但行为保持不变，直到后续 PR 启用新匹配路径。测试覆盖较完整，核心逻辑变更经过 12 条 review 评论推敲。 - 风险标记：调用方假设 block_hash != None 为全块，反向映射集合可能增加内存压力，部分注册在并发下的状态一致性，BlockHashListWithBlockSize 视图可能陈旧

关联脉络

- PR #43468 [feature][kv_offload] Self-describing KV events for OffloadingConnector: 该 PR 引入了 KV 事件（BlockStored/BlockRemoved）框架，本 PR 的 cache_partial_block 依赖这些事件来通知 offloading 等子系统。