PR #46384 完整报告

vllm-project/vllm

[2/N][Core] support partial prefix cache hit for hybrid model

合并时间：2026-07-12 13:37

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46384

执行摘要

- 一句话：支持混合模型细粒度前缀缓存命中
- 推荐动作：此 PR 设计精良，值得仔细阅读，特别是 _mamba_block_aligned_split 的分片逻辑和 CoW 两种路径。对于需要为自定义混合模型添加类似支持的工程师有很高参考价值。讨论中涉及的遗留问题（SWA、释放时机、拷贝顺序）需要在后续 PR 中跟踪。

功能与动机

此前前缀缓存只能按完整物理块命中，对于 Mamba 块大小比 Full Attention 块大的混合模型（例如 Mamba 512、Full Attention 128、哈希粒度 16），若公共前缀结束于 Mamba 块内部则缓存完全失效。本 PR 通过引入 prefix_match_unit 和显式 token 长度跟踪，即使物理块未对齐也能部分命中缓存，显著降低多轮对话延迟。详细动机见 RFC #45702。

实现拆解

1. 新增 prefix_match_unit 配置：在 CacheConfig 中增加 prefix_match_unit 字段（原 hash_block_size），决定哈希计算粒度。resolve_kv_cache_block_sizes 从 prefix_match_unit 导出 hash_block_size；若未指定则取各组 block_size 的 GCD。涉及文件：vllm/config/cache.py、vllm/v1/core/kv_cache_utils.py。
2. 缓存管理器支持细粒度命中：在 SingleTypeKVCacheManager 添加类变量 supports_fine_grained_hash_lookup。FullAttentionManager 和 MambaManager 覆写 find_longest_cache_hit，返回 (blocks, hit_length)。MambaManager 利用细粒度哈希在物理块内查找状态；FullAttentionManager 先匹配完整块，再在下一块内哈希边界探索。涉及文件：vllm/v1/core/single_type_kv_cache_manager.py、vllm/v1/core/kv_cache_coordinator.py。
3. 混合协调器收敛：HybridCoordinator 计算各组的 hit_length 后取最小值作为公共前缀长度。Full Attention 的命中是向下闭合的，直接 trim 而不触发二次查询。涉及文件：vllm/v1/core/kv_cache_coordinator.py。
4. 调度器分片：当 mamba_partial_cache_hit 启用时，_mamba_block_aligned_split 在提示词末尾添加一次哈希边界停止，使得 Mamba 能注册精确循环状态；恢复执行时第一个分片停在下一个物理块边界。涉及文件：vllm/v1/core/sched/scheduler.py。
5. 写时复制（CoW）：两种 CoW 路径——新请求部分命中时分配私有块并复制旧块内容；正在运行的请求注册部分快照时创建缓存专用块并移入快照。管理器队列收集 (源块, 目标块)

对，通过 SchedulerOutput 下发 KVCacheBlockCopy。涉及文件：vllm/v1/core/single_type_kv_cache_manager.py、vllm/v1/core/block_pool.py、vllm/v1/core/kv_cache_utils.py。

6. Worker 端拷贝与生命周期：copy_kv_cache_blocks_inplace 利用 async_tensor_h2d 异步下发拷贝命令，在零化新块后、模型前向前执行，去重各层的存储。拷贝两端通过 deferred_frees 延迟释放。涉及文件：vllm/v1/worker/utils.py、vllm/v1/core/sched/scheduler.py。
7. 测试配套：新增 tests/v1/core/prefix_cache/test_partial_prefix_cache_hits.py，覆盖调度分片、部分命中查找、CoW 场景；修改 tests/v1/core/test_kv_cache_utils.py 和 tests/v1/core/test_deferred_block_free.py。

关键文件：

- vllm/v1/core/single_type_kv_cache_manager.py（模块 缓存管理器；类别 source；类型 core-logic；符号 _has_partial_local_hit, take_pending_cow_copies, _apply_cow, cache_blocks）：核心缓存管理器基类，新增 partial hit 检测、CoW 书签、Cache 尾部块等方法，是最关键的实现文件。
- tests/v1/core/prefix_cache/test_partial_prefix_cache_hits.py（模块 测试；类别 test；类型 test-coverage；符号 _auto_init_hash_fn, test_mamba_align_split_partial_tail_schedule, test_hybrid_mamba_align_partial_hash_hit, test_hybrid_mamba_partial_tail_owner_uses_cow_on_continue）：新增 816 行测试覆盖所有关键场景：调度分片、部分命中查找、CoW 行为，是验证实现正确性的核心测试文件。
- vllm/v1/core/kv_cache_coordinator.py（模块 协调器；类别 source；类型 core-logic；符号 _cache_hit_alignment_tokens, _get_block_hashes）：混合协调器新增 enable_partial_hash_hits 和 _cache_hit_alignment_tokens，实现各组命中长度收敛，并传播显式 hit_length。
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _free_cow_retained_blocks）：调度器新增 mamba_partial_cache_hit 标志，修改 _mamba_block_aligned_split 以在哈希边界注册部分尾部状态，并添加 CoW 块释放机制。
- vllm/v1/core/kv_cache_utils.py（模块 缓存工具；类别 source；类型 core-logic；符号 KVCacheBlockCopy）：新增 KVCacheBlockCopy NamedTuple 和 resolve_block_hashes 工具的细粒度版本，修改 resolve_kv_cache_block_sizes 配置项。
- vllm/v1/worker/utils.py（模块 工作器；类别 source；类型 core-logic；符号 copy_kv_cache_blocks_inplace）：新增 copy_kv_cache_blocks_inplace 函数，负责在 GPU 上执行块拷贝，是 CoW 落地到硬件的关键。
- vllm/v1/core/block_pool.py（模块 块池；类别 source；类型 core-logic；符号 move_block_hashes）：新增 move_block_hashes 方法，用于将部分尾部条目的哈希从一个块迁移到另一个块，支持 CoW 第二种路径。
- vllm/v1/core/kv_cache_manager.py（模块 缓存管理器；类别 source；类型 core-logic；符号 take_kv_cache_block_copies）：顶层 KVCacheManager 新增 take_kv_cache_block_copies 接口，聚合各管理器队列的 KVCacheBlockCopy 列表并传递给调度输出。

关键符号：_has_partial_local_hit, take_pending_cow_copies, _apply_cow, _cache_partial_tail_block, _cache_hit_alignment_tokens, _mamba_block_aligned_split, _free_cow_retained_blocks, copy_kv_cache_blocks_inplace, move_block_hashes, take_kv_cache_block_copies

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

核心缓存管理器基类，新增 partial hit 检测、CoW 书签、Cache 尾部块等方法，是最关键的实现文件。

```
def _has_partial_local_hit(
    self,
    new_computed_blocks: Sequence[KVCacheBlock],
    num_local_computed_tokens: int,
) -> bool:
    # 检查本地命中是否结束在块内部（非块尾）
    # 如果命中结尾不在块边界，则需要 CoW 来保护共享缓存区块
    return (
        len(new_computed_blocks) > 0
        and num_local_computed_tokens % self.block_size != 0
    )

def get_num_blocks_to_allocate(
    self,
    request_id: str,
    num_tokens: int,
    new_computed_blocks: Sequence[KVCacheBlock],
    total_computed_tokens: int,
    num_local_computed_tokens: int,
    num_tokens_main_model: int,
    apply_admission_cap: bool = False,
) -> int:
    ... # 前置计算
    if self._has_partial_local_hit(new_computed_blocks, num_local_computed_tokens):
        # 为部分命中 CoW 重定向预留一个额外块
        num_new_blocks += 1
    return num_new_blocks + num_evictable_blocks
```

tests/v1/core/prefix_cache/test_partial_prefix_cache_hits.py

新增 816 行测试覆盖所有关键场景：调度分片、部分命中查找、CoW 行为，是验证实现正确性的核心测试文件。

```
def test_mamba_align_split_partial_tail_schedule():
    """Chunk ends with partial hits: block-aligned chunks, one extra stop
    at last hash boundary, then remaining tokens. block=512, hash=32, prompt=10000."""
    block_size = 512
    hash_block_size = 32
```

```
mock = SimpleNamespace(
    cache_config=SimpleNamespace(block_size=block_size),
    use_eagle=False,
    hash_block_size=hash_block_size,
    mamba_partial_cache_hit=True,
)
split = Scheduler._mamba_block_aligned_split
req = make_request("0", [0] * 10000, hash_block_size, sha256)
req.num_computed_tokens = 0
assert split(self=mock, request=req, num_new_tokens=8192) == 8192
req.num_computed_tokens = 8192
assert split(self=mock, request=req, num_new_tokens=1808) == 1536 # 停在下一块边界
req.num_computed_tokens = 9728
assert split(self=mock, request=req, num_new_tokens=272) == 256 # 停在最后的哈希边界
req.num_computed_tokens = 9984
assert split(self=mock, request=req, num_new_tokens=16) == 16 # 最后块无需停止
```

评论区精华

- SWA 部分命中支持：ivanium 提出是否保留 Sliding Window Attention 的 partial hit 代码，认为可能成为技术债。最终决定 defer 到后续 PR（#47782 可能重新设计）。
- 释放过早风险：njhill 指出在异步调度情况下 CoW 块的释放可能过早，需要类似 #47728 的修复。通过 rebase #47728 解决。
- CoW 拷贝顺序依赖：Codex 审查指出当快照拷贝和 CoW 拷贝在同一调度步骤时，若无依赖顺序可能导致读取旧数据。开发者未明确修复，但最终实现将所有拷贝收集后批量提交，风险仍存。
- SWA 部分命中支持 (design): 决定 defer 到后续 PR，当前仅支持 Full Attention + Mamba 混合。
- CoW 块释放过早 (correctness): 通过 rebase #47728，利用 num_in_flight_tokens 保护释放时机。
- CoW 拷贝顺序依赖 (correctness): 开发者未显式排序，但最终实现将所有拷贝收集后批量提交，风险尚未完全消除。

风险与影响

- 风险：
  1. 正确性风险：copy_kv_cache_blocks_inplace 未保证拷贝顺序，若同一调度步骤中存在依赖（快照→CoW），可能读取未更新的源数据（Codex P1）。
  2. 兼容性风险：拷贝函数假设 block-major 布局，对非 block-major 后端（如 ROCm）可能错误；虽添加断言，但可能仍需验证。
  3. 内存泄漏风险：partial-hit bookkeeping 在 deferred free 路径中未清理 num_cached_hash_block 残留，可能影响后续请求的缓存写入（Codex P2）。
  4. 性能风险：CoW 拷贝增加预填充延迟，但仅涉及尾部块，影响有限。

5. 配置风险：若 prefix_match_unit 不能整除各组 block_size，resolve_kv_cache_block_sizes 会抛出 ValueError。

- 影响：
  - 用户影响：Hybrid 模型（如 Qwen3.5）多轮对话 second latency 降低约 28%（0.134s → 0.096s），且无需修改模型。
  - 系统影响：新增 SchedulerOutput.kv_cache_block_copies 字段，worker 端新增拷贝步骤；调度器分裂逻辑更复杂，但向后兼容（不启用 prefix_match_unit 时无行为变化）。
  - 团队影响：需要维护 CoW 状态一致性及与 #47782 的协同；建议为自定义模型集成该特性时参考。
  - 风险标记：CoW 拷贝顺序依赖，block-major 布局假设，partial-hit 书签泄漏，异步释放时机

关联脉络

- PR #44455 [2/N][KV-Cache Layout Refactor] Pack K/V into the content dim across attention backends: 同一系列 KV 缓存重构，该 PR 将块布局统一为 block-major，为本 PR 的 CoW 拷贝提供了基础假设。
- PR #47314 [BugFix] Fix packed HND KV cache reshape for FlashAttention: 同一模块的核心 bug 修复，确保 block-major 布局正确，与此 PR 的块拷贝假设密切相关。