PR #42258 完整报告

vllm-project/vllm

[Core][DSV4] Skip caching SWA blocks that can never serve a prefix-cache hit

合并时间：2026-05-15 15:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42258

执行摘要

- 一句话：跳过无法命中 Prefix Cache 的 SWA 块
- 推荐动作：值得精读，设计模式（通过 mask 避免无效缓存）可供类似场景借鉴。但需关注 review 中提出的共享物理块断言风险和事件过滤问题，建议在后续 PR 中验证并修复可能的问题。

功能与动机

DeepSeek-V4 的 full-attention 层与 SWA 层使用不同 block size（256 vs 64/8/4），SWA 的 find_longest_cache_hit 仅返回每个 lcm 对齐段内靠后的部分块，早先的块永远无法被命中。缓存这些块会污染 prefix-cache hash map 并占用 LRU 列表，因此需要在缓存时跳过它们。

实现拆解

1. 在 HybridKVCacheCoordinator 中新增 cache_blocks 方法，将 num_computed_tokens 对齐到 lcm_block_size，并传递 alignment_tokens=self.lcm_block_size 给各 single_type_manager。
2. 在 KVCacheManager.cache_blocks 中加入 alignment_tokens 参数。若该值大于 block_size，则调用 _cache_block_mask 生成 block_mask，否则走快速路径（mask=None）。
3. SWAManager._cache_block_mask 计算对齐段大小 per_segment = alignment_tokens // block_size 和尾窗长度 tail = ceil((sliding_window - 1) / block_size)。当 tail < per_segment 时，每个段内前 skip = per_segment - tail 个块将被跳过（mask=False），只有最后 tail 个块可缓存。
4. BlockPool.cache_full_blocks 新增可选参数 block_mask。在遍历新块时，如果块被标记为跳过（block_mask[i] == False），则跳过缓存，就像处理 null 块一样。同时确保 event 的 token_ids 和 extra_keys_list 也按 mask 过滤（但 review 指出可能仍有问题，需关注）。
5. 新增两个单元测试，验证 SWA 尾窗限制和 lcm 边界截断行为。

关键文件：

- vllm/v1/core/single_type_kv_cache_manager.py（模块 缓存核心；类别 source；类型 core-logic；符号 cache_blocks, _cache_block_mask）：核心实现，新增 cache_blocks 的 alignment_tokens 参数、默认 _cache_block_mask 以及 SWAManager 的覆盖实现，是跳过不可达块的关键逻辑所在。

- vllm/v1/core/block_pool.py（模块 缓存块池；类别 source；类型 core-logic；符号 cache_full_blocks）：缓存引擎底层，cache_full_blocks 新增 block_mask 参数，在遍历块时跳过被 mask 标记的块，是跳过缓存的具体执行层。
- vllm/v1/core/kv_cache_coordinator.py（模块 协调器；类别 source；类型 core-logic；符号 cache_blocks）：协调器层，HybridKVCacheCoordinator 新增 cache_blocks 方法，对齐 num_computed_tokens 到 lcm_block_size 并传递 alignment_tokens 给各 manager，是触发 mask 链路的入口。
- tests/v1/core/test_prefix_caching.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_hybrid_cache_blocks_swa_tail_window_only, test_hybrid_cache_blocks_clamped_to_lcm）：新增两个测试函数，分别验证 SWA 尾窗限制（仅尾段块被缓存）和 lcm 边界截断（最后一段之外不缓存）的正确性。

关键符号：cache_blocks, _cache_block_mask, cache_full_blocks

关键源码片段

vllm/v1/core/single_type_kv_cache_manager.py

核心实现，新增 cache_blocks 的 alignment_tokens 参数、默认 _cache_block_mask 以及 SWAManager 的覆盖实现，是跳过不可达块的关键逻辑所在。

```
# KVCacheManager.cache_blocks: 新增 alignment_tokens 参数
def cache_blocks(
    self,
    request: Request,
    num_tokens: int,
    alignment_tokens: int | None = None,
) -> None:
    # 当 alignment_tokens > block_size 时，需要根据缓存命中对齐要求
    # 生成 block_mask 以跳过不可能被命中的块
    if alignment_tokens is None or alignment_tokens <= self.block_size:
        block_mask = None # 快速路径：不需要 mask
    else:
        block_mask = self._cache_block_mask(
            num_cached_blocks, num_full_blocks, alignment_tokens
        )
    self.block_pool.cache_full_blocks(
        request=request,
        blocks=self.req_to_blocks[request.request_id],
        num_cached_blocks=num_cached_blocks,
        num_full_blocks=num_full_blocks,
        block_size=self.block_size,
        kv_cache_group_id=self.kv_cache_group_id,
        block_mask=block_mask, # 传递 mask
    )

# 基类默认实现：所有块都可缓存（返回 None）
def _cache_block_mask(
```

```
        self,
        num_cached_blocks: int,
        num_full_blocks: int,
        alignment_tokens: int,
    ) -> list[bool] | None:
        return None

    # SWAManager 覆盖：只保留每个 lcm 段内的尾窗块
    def _cache_block_mask(
        self, num_cached_blocks: int, num_full_blocks: int, alignment_tokens: int
    ) -> list[bool] | None:
        assert alignment_tokens > self.block_size
        per_segment = alignment_tokens // self.block_size
        tail = cdiv(self.sliding_window - 1, self.block_size)
        if tail >= per_segment:
            return None # 尾窗覆盖整个段，无需跳过
        skip = per_segment - tail
        return [
            i % per_segment >= skip for i in range(num_cached_blocks, num_full_blocks)
        ]
```

vllm/v1/core/block_pool.py

缓存引擎底层，cache_full_blocks 新增 block_mask 参数，在遍历块时跳过被 mask 标记的块，是跳过缓存的具体执行层。

```
    def cache_full_blocks(
        self,
        request: Request,
        blocks: list[KVCacheBlock],
        num_cached_blocks: int,
        num_full_blocks: int,
        block_size: int,
        kv_cache_group_id: int,
        block_mask: list[bool] | None = None, # 新增参数
    ) -> None:
        if num_cached_blocks >= num_full_blocks:
            return
        new_full_blocks = blocks[num_cached_blocks:num_full_blocks]
        # block_mask 长度必须等于新块数量
        assert block_mask is None or len(block_mask) == len(new_full_blocks)

        # ...（省略 block_hashes 计算）

        for i, blk in enumerate(new_full_blocks):
            # 跳过 null 块或被 mask 标记为 False 的块
            if blk.is_null or (block_mask is not None and not block_mask[i]):
                continue
            # 缓存逻辑：设置 block_hash，插入到 hash map
            block_hash = new_block_hashes[i]
```

```
block_hash_with_group_id = make_block_hash_with_group_id(
    block_hash, kv_cache_group_id
)
blk.block_hash = block_hash_with_group_id
self.cached_block_hash_to_block.insert(block_hash_with_group_id, blk)
```

评论区精华

Review 中 gemini-code-assist[bot] 提出了两个高优先级问题：(1) 共享物理块可能导致断言失败和 map 泄漏；(2) BlockStored 事件中 token_ids 未过滤。ivanium 对问题 (1) 回复 'Not true.' 但未提供详细论证。问题 (2) 未获直接回应。此外，建议增加共享物理块场景的测试覆盖也未在合并版本中见及。这些未解决的点可能仍存在隐患。

- 共享物理块断言与 map 泄漏风险 (correctness): ivanium 回复 'Not true.'。未进一步澄清。PR 已合并，但风险未完全消除。
- BlockStored 事件 token_ids 未过滤 (correctness): 未获作者回应。PR 已合并，但问题未修复。
- 缺少共享物理块场景的测试覆盖 (testing): PR 中未增加此类测试。可能作者认为无必要。

风险与影响

- 风险：非混合模式（alignment_tokens=None）走快速路径，回归风险低。但对混合模型，若 _cache_block_mask 计算错误可能导致合法缓存被跳过或非法缓存被保留。Review 中指出共享物理块可能触发断言（assert blk.block_hash is None），若存在两个组共享同一物理块且都尝试缓存，则第二个组会看到 non-None block_hash 导致断言失败。此外，BlockStored 事件中 token_ids 未按 block_mask 过滤，可能导致下游分布式缓存不匹配。建议后续验证。
- 影响：仅影响使用 HybridKVCacheCoordinator 的模型（如 DeepSeek-V4），减少 prefix-cache 条目数，提高缓存效率。对其他模型无影响。变动涉及核心缓存路径，但配合了快速路径开关，风险可控。
- 风险标记：共享块断言风险，事件过滤缺失

关联脉络

- PR #42444 [Model Runner V2][Bug Fix][DSV4] Ensure lazy attention state initializations happen during cudagraph capture: 同为 DeepSeek-V4 相关修复，涉及模型运行器 v2 的 CUDA Graph 初始化问题。
- PR #41986 [Bugfix] Add swiglu limits to deepgemm fp8 methods: 同为 DeepSeek-V4 系列，涉及 FP8 MoE 的 SwiGLU 截断修复。