PR #52482 完整报告

vllm-project/vllm

[Bugfix][V1][Multimodal] Ignore stale same-step encoder cache evictions

合并时间：2026-08-17 06:21

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52482

执行摘要

- 一句话：忽略同一步调度内过期的 encoder 缓存驱逐通知，修复引擎崩溃
- 推荐动作：值得精读：这是理解 V1 调度器 encoder cache 生命周期（schedule → can_allocate → 驱逐通知 → worker tensor 回收）的最小闭环案例，修复点小而关键，回归测试精确构造了竞态场景。可借鉴的设计决策是「用最终状态过滤过期事件」：系统在两个时点之间先删除后重建同一资源时，不应把中间态事件外发。建议合入后持续跟踪 Issue #38551，若在压测中仍复现 Encoder cache miss，下一步应在调度器层约束 MTP 草稿提议对 encoder cache 的引用。

功能与动机

PR body 明确指出根因："An entry can be evicted early in schedule() and added to self.freed, then allocated again later in the same scheduling pass. Previously, the stale eviction was still sent to the model runner, which removed the tensor before _gather_mm_embeddings() used it. On current main, this terminates the engine with: RuntimeError: Encoder cache miss for ." 关联 Issue #38551 记录了生产环境的完整表现：MTP 投机解码 + 多模态输入 + 高并发下，engine 在 propose_draft_token_ids → _gather_mm_embeddings 路径触发 assert encoder_output is not None 崩溃并丢弃全部在途请求；issue 给出的临时 workaround 是把致命断言改为 check-and-skip。本 PR 从缓存管理侧消除了其中一类误报驱逐，但没有改变跨 pass 的真实驱逐行为。

实现拆解

1. 根因定位：调度 pass 内 freed 通知的竞态窗口

EncoderCacheManager（vllm/v1/core/encoder_cache_manager.py）在 can_allocate() 驱逐条目时把 mm_hash 记入 self.freed，调度结束时由 get_freed_mm_hashes() 一次性取走并清空，用于通知 worker 删除对应 encoder tensor。问题在于：同一 pass 内该 mm_hash 可能因另一个请求引用同一多模态输入（或相同 mm hash）而被再次 allocate() 回 cached，此时 freed 中的记录已过期，但旧实现仍原样返回，worker 侧 tensor 被提前删除，最终在 _gather_mm_embeddings() 处崩溃。

2. 核心修复：用最终 cached 状态过滤 freed 记录

将 get_freed_mm_hashes() 的返回值从 self.freed 改为列表推导式 [mm_hash for mm_hash in self.freed if mm_hash not in self.cached]：只有最终仍处于被驱逐状态（不在

cached）的条目才上报给 worker。self.freed 依然整体清空，freeable 与 num_freeable_slots 等缓存记账逻辑完全不变，实际驱逐行为不变，只是不再外发错误事件。

3. 回归测试：构造驱逐后重分配场景

在 tests/v1/core/test_encoder_cache_manager.py 新增 test_reallocated_hash_is_not_reported_as_freed：用 cache_size=8 与三个各占 4 槽的请求 a、b、c，先由 a、b 占满缓存并释放，再分配 c 触发驱逐使 a 与 b 进入 freed 记录，随后在同一 pass 内重新分配 a，最后断言 get_freed_mm_hashes() == ["b"]。该测试在 main 上失败（旧逻辑返回 ['a', 'b']），修复后通过。

4. 配套与验证

无配置、schema 或部署配套变更。验证方式：pytest tests/v1/core/test_encoder_cache_manager.py 17 个用例全部通过；pre-commit 首轮失败后按提示修复并通过。提交历史共 3 个 commit，其中 2 个为合并 main 的同步提交，核心改动集中在首个 commit。

关键文件：

- vllm/v1/core/encoder_cache_manager.py（模块 缓存管理；类别 source；类型 core-logic；符号 get_freed_mm_hashes）：核心修复文件。get_freed_mm_hashes() 改为只返回最终不在 self.cached 中的驱逐记录，消除同一步调度内驱逐后重新分配导致的过期 freed 通知，避免 worker 误删仍被使用的 encoder tensor。
- tests/v1/core/test_encoder_cache_manager.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_reallocated_hash_is_not_reported_as_freed）：回归测试文件。新增 test_reallocated_hash_is_not_reported_as_freed，构造同一调度 pass 内驱逐后重分配的场景，在 main 上失败（返回 ['a', 'b']）、修复后通过，固化竞态修复行为。

关键符号：get_freed_mm_hashes, test_reallocated_hash_is_not_reported_as_freed

关键源码片段

vllm/v1/core/encoder_cache_manager.py

核心修复文件。get_freed_mm_hashes() 改为只返回最终不在 self.cached 中的驱逐记录，消除同一步调度内驱逐后重新分配导致的过期 freed 通知，避免 worker 误删仍被使用的 encoder tensor。

```
def get_freed_mm_hashes(self) -> list[str]:
    """获取并清空最近被驱逐的 encoder cache 条目列表。

    返回值由 scheduler 用于通知 worker：哪些 encoder 输出需要从
    worker 侧缓存中删除；内部列表在调用后被清空。
    """
    # 修复要点：同一个调度 pass 内，某个 mm_hash 可能先被驱逐
    # （加入 self.freed），随后又在同一 pass 后期被重新分配，
    # 重新进入 self.cached。旧逻辑原样返回 freed 列表，
    # 会使 worker 提前删除该 tensor，导致稍后
    # _gather_mm_embeddings() 触发 "Encoder cache miss" 崩溃。
    # 这里只返回最终不在 self.cached 中的条目，被重新分配的
```

```
    # 条目对应 tensor 得以在 worker 侧保留。
    freed = [mm_hash for mm_hash in self.freed if mm_hash not in self.cached]
    self.freed = []
    return freed
```

tests/v1/core/test_encoder_cache_manager.py

回归测试文件。新增 test_reallocated_hash_is_not_reported_as_freed，构造同一调度 pass 内驱逐后重分配的场景，在 main 上失败（返回 ['a', 'b']）、修复后通过，固化竞态修复行为。

```
def test_reallocated_hash_is_not_reported_as_freed():
    """回归测试：同一调度 pass 内被驱逐后又重新分配的 mm_hash
    不应被报告为 freed，否则 worker 会删除仍在使用的 tensor。
    """
    manager = EncoderCacheManager(cache_size=8)
    req_a = MockRequest("reqA", ["a"], [4])
    req_b = MockRequest("reqB", ["b"], [4])
    req_c = MockRequest("reqC", ["c"], [4])

    # 先由 a、b 占满 8 个槽位，再释放它们使其进入 freeable。
    manager.allocate(req_a, 0)
    manager.allocate(req_b, 0)
    manager.free(req_a)
    manager.free(req_b)

    # 分配 c 触发驱逐，a 与 b 均被记录进 self.freed。
    assert manager.can_allocate(req_c, 0, int(1e9), 0)
    manager.allocate(req_c, 0)

    # 同一调度 pass 内 a 再次被分配回 cached，
    # 因此 get_freed_mm_hashes() 必须排除 a，只保留 b。
    assert manager.can_allocate(req_a, 0, int(1e9), 0)
    manager.allocate(req_a, 0)

    assert manager.get_freed_mm_hashes() == ["b"]
```

评论区精华

本轮 review 几乎没有实质性技术交锋。PR 的 5 条评论全部是 CI 触发（/ci run）与 pre-commit 失败提示，review comments 为 0。claude[bot] 自动说明："This pull request is from a fork — automated review is disabled."，即 fork 来源 PR 默认跳过自动审查。维护者 Isotr0py 在 CI 绿后直接 APPROVED，说明修复意图与边界在提交时已经清晰。唯一值得一提的过程是 pre-commit 首轮失败、按提示修复后通过，无遗留未解决问题。

- fork PR 的自动 review 状态 (other): 未触发一次性 review，维护者 Isotr0py 直接 APPROVED。

风险与影响

- 风险：

1. 语义依赖：修复正确性依赖 get_freed_mm_hashes() 的调用时机——它必须在整个调度 pass 结束后、cached 状态不再被同 pass 修改时调用。当前 V1 调度流程满足该前提；若未来在 pass 中途调用，过滤可能漏报真正被驱逐的条目，最坏情况是 worker 侧 tensor 滞留（内存延迟回收）而非崩溃，危害较低。
2. 覆盖范围有限：本 PR 只消除同 pass 驱逐后重分配这一类误报。#38551 描述的完整故障（MTP 草稿提议引用更早 pass 被驱逐的条目）未被覆盖，issue 仍为 open；生产压力下若仍复现，需要调度器侧 gating MTP proposal 或禁止驱逐仍被引用的条目。
3. 回归风险：最坏情况是漏掉一条 freed 通知导致 worker 侧 tensor 延迟回收，不会引入崩溃；freeable 与 num_freeable_slots 记账未动，内存账本保持一致。
4. 测试覆盖：新增测试覆盖了单次重分配场景，但未覆盖同一 pass 内被驱逐两次、或与 MTP 草稿线程组合的集成场景。 - 影响：用户影响：多模态（图片 / 视频）+ 高并发、尤其 MTP 投机解码场景下，引擎不再因 Encoder cache miss 整体掉线，在途请求不再被意外丢弃。系统影响：get_freed_mm_hashes() 增加一次 O(n) 过滤遍历，n 为驱逐记录数，性能影响可忽略；worker 侧 tensor 生命周期更贴近最终缓存状态，减少不必要的删除与重建。团队影响：为 #38551 提供了部分缓解与一个可复用的回归测试范式，后续调度器级修复可以此测试为基线。 - 风险标记：调度器核心路径变更，仅修复同 pass 竞态窗口，关联 issue #38551 仍 open，依赖 cached 与 freed 状态一致性

关联脉络

- PR #38551 [Bug]: AssertionError: Encoder cache miss crashes engine with MTP + multimodal under high concurrency: 直接关联 Issue（非历史 PR）：描述同一 Encoder cache miss 崩溃场景与生产复现条件；本 PR 仅修复其中「同 pass 驱逐后重分配」的误报子场景，完整问题仍 open。