# PR #45357 完整报告

vllm-project/vllm

[Bugfix] Defer block freeing until in-flight steps finish under async scheduling + PD KV consumer

合并时间：2026-06-16 05:36

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45357

## 执行摘要

- 一句话：延迟块释放修复异步调度与 PD 分解竞态
- 推荐动作：推荐使用 PD 分解的团队立即合并此修复。PR 的设计模式（步骤栅栏 + 延迟释放）值得学习，展示了如何在不破坏现有逻辑的前提下解决复杂的并发竞态。核心贡献者阅读可关注 pop_blocks_for_free 的设计和延迟释放队列的条件判断。

## 功能与动机

异步调度下，调度器可能在步骤 N 输出被处理前就调度步骤 N+1。若请求在步骤 N 输出后完成或抢占，其 KV 块被立即释放，但步骤 N+1 仍可能写入这些块。若新 PD 消费者请求重用这些块，NIC/RDMA 写入的接收 KV 与过时 GPU 写入未排序，导致数据损坏。本 PR 延迟块归还以修复此竞态。

## 实现拆解

1. 引入步骤序列号栅栏：在 Scheduler 中增加 sched_step_seq 和 processed_step_seq，SchedulerOutput 携带 sched_seq。非空步骤调度时递增 sched_seq；update_from_output 处理输出时更新 processed_step_seq。
2. 拆分缓存释放逻辑：在 KV 缓存管理器层新增 pop_blocks_for_free，仅移除请求的 bookkeeping 并返回块列表，不归还 block pool。原有的 free 改为调用 pop_blocks_for_free 后立即归还。
3. 延迟释放队列：Scheduler 维护 _deferred_frees: deque[(fence_seq, blocks)]，只有当 processed_step_seq >= fence_seq 时才能安全归还。
4. 调度器释放路径：统一的 _free_request_blocks 方法。若延迟启用且存在在调度步骤，则调用 pop_blocks_for_free 并将块入队；否则直接 free。
5. 排空延迟队列：update_from_output 中每次处理后调用 _drain_deferred_frees，将满足条件的块通过 block_pool.free_blocks 归还。
6. 启用条件：仅当 max_concurrent_batches > 1（异步调度或 PP）且 KV 传输配置为消费者时设置 defer_block_free = True。

关键文件：

- tests/v1/core/test_deferred_block_free.py（模块 测试覆盖；类别 test；类型 test-coverage；符号 _make_model_runner_output, _create_deferring_scheduler,

_setup_request_with_inflight_step, test_gate_enabled_for_async_consumer）：新增完整测试套件，覆盖延迟释放的启用条件、正常推迟、无在调步骤立即释放、abort 场景等。

- 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 free, pop_blocks_for_free）：核心拆分：新增 pop_blocks_for_free 方法，使调用方可以延迟归还 block pool。原有 free 方法基于 pop_blocks_for_free 实现。
- vllm/v1/core/kv_cache_coordinator.py（模块 缓存协调器；类别 source；类型 core-logic；符号 pop_blocks_for_free）：新增 pop_blocks_for_free 方法，将单个请求的 block 收集逻辑委托给所有单类型管理器。
- vllm/v1/core/kv_cache_manager.py（模块 缓存管理层；类别 source；类型 core-logic；符号 pop_blocks_for_free）：在顶层 KVCacheManager 中新增 pop_blocks_for_free 方法，将请求传递给协调器。
- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _free_request_blocks, _drain_deferred_frees）：核心调度逻辑变更：引入 defer_block_free 门控、步骤序列号、延迟释放队列、_free_request_blocks 和 _drain_deferred_frees 方法。

关键符号：_free_request_blocks, _drain_deferred_frees, pop_blocks_for_free, free, schedule, update_from_output

## 关键源码片段

### tests/v1/core/test_deferred_block_free.py

新增完整测试套件，覆盖延迟释放的启用条件、正常推迟、无在调步骤立即释放、abort 场景等。

```
def _create_deferring_scheduler():
    """创建强制启用延迟释放的调度器（生产环境还需 KV 消费者）"""
    # 直接设置 defer_block_free 为 True，绕过生产环境的条件判断
    scheduler = create_scheduler(model=MODEL, async_scheduling=True)
    scheduler.defer_block_free = True
    return scheduler


def _setup_request_with_inflight_step(scheduler, max_tokens: int = 5):
    """
    模拟一个请求的 prefill（步骤1）和一次 speculatively 调度的 decode（步骤2），
    用于测试延迟释放场景。返回 (request, out0, out1)。
    """
    request = create_requests(
        num_requests=1,
        num_tokens=NUM_PROMPT_TOKENS, # 33 tokens, block_size=16 时占用 3 个 block
        max_tokens=max_tokens,
        stop_token_ids=[STOP_TOKEN_ID],
    )[0]
    scheduler.add_request(request)
    out0 = scheduler.schedule() # 步骤 1：prefill
```

```
    assert out0.num_scheduled_tokens[request.request_id] == NUM_PROMPT_TOKENS
    out1 = scheduler.schedule() # 步骤 2：decode（提前调度）
    assert out1.num_scheduled_tokens[request.request_id] == 1
    return request, out0, out1
```

### vllm/v1/core/single_type_kv_cache_manager.py

核心拆分：新增 pop_blocks_for_free 方法，使调用方可以延迟归还 block pool。原有 free 方法基于 pop_blocks_for_free 实现。

```
# 在 SingleTypeKVCacheManager 中新增的方法
def pop_blocks_for_free(self, request_id: str) -> list[KVCacheBlock]:
    """
    弹出请求的 bookkeeping 并返回其 block 列表，但 *不* 归还给 block pool。
    调用方必须后续通过 block_pool.free_blocks 归还 block，顺序须为逆序。
    """
    # 默认返回 []，防止请求在被分配 block 前就被释放（abort）
    req_blocks = self.req_to_blocks.pop(request_id, [])
    # 同时清理缓存的 block 计数
    self.num_cached_block.pop(request_id, None)
    return req_blocks


def free(self, request_id: str) -> None:
    """
    立即释放请求的所有 block（逆序归还给 block pool）。
    现在基于 pop_blocks_for_free 实现，保持原有行为不变。
    """
    # 逆序释放，使尾部 block 优先回池
    self.block_pool.free_blocks(reversed(self.pop_blocks_for_free(request_id)))
```

## 评论区精华

Review 中主要讨论包括：NickLucche 质疑添加日志的必要性，njhill 回复保留日志是担心性能影响，但也可移除；njhill 提供了简化实现并补充了测试用例；ZJY0516 要求提供精度测试结果，lIx-08 后提供了 GSM8K 基准测试结果确认正确性。后续 njhill 还追加了覆盖 PP 场景的提交。

- 延迟释放启用时的日志级别 (design): 最终保留了 info 日志，但后续提交已移除无用日志。
- 精度测试验证 (testing): 精度测试通过，覆盖提升。

## 风险与影响

- 风险：延迟释放可能在高 KV 缓存压力下降低池利用率，但该功能仅在有异步调度和 PD KV 消费者时启用，降低了影响范围。新引入的序列号逻辑依赖 update_from_output 按 FIFO 顺序调用，若有变更可能导致不一致。pop_blocks_for_free 与 free 的分离需要确保所有调用方都正确迁移，避免双重释放或遗漏释放。此修复不涉及模型侧变更，风险可控。
- 影响：影响范围仅限于启用异步调度且使用 PD KV 消费者（分解部署）的场景。普通用户不受影响。对于使用 PD 分解的团队，此修复至关重要，能消除因竞态导致的 KV 损坏和错误

输出。系统在启用后会有较小的额外计算开销（检查序列号和队列操作），但可忽略。测试覆盖包括专门的单元测试和已在 PD 环境中验证的精度测试。

- 风险标记：核心路径变更，并发竞态修复，KV 缓存管理变更，需要精度验证，启用条件有限

## 关联脉络

- PR #45096 [Bugfix] Alternative approach to defer block free（从评论推断）：本 PR 是 #45096 的替代实现，讨论中提及。
- PR #47373 [Bugfix] Fix zero_blocks overwriting KV cache data received via RDMA：由 llx-08 在讨论中提及，是同一功能线上的相关修复。