PR #48481 完整报告

vllm-project/vllm

[KV Connector] Fix PD async scheduling race condition for hybrid attn models

合并时间：2026-07-16 18:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48481

执行摘要

- 一句话：修复混合模型 PD 异步调度精度回归
- 推荐动作：建议精读此 PR，特别是调度器侧跳过清零的设计模式和使用外部令牌授权推导覆盖块的方法。值得注意的设计决策：避免在异步管道中加入 CUDA 同步点，而是通过元数据告知调度器排除特定块。

功能与动机

PR 意图修复两个已知 bug：TP>1 时 PD 异步调度精度崩溃（issue #37285）和 TP==1 时 Qwen3.5 精度塌陷（issue #42182）。同时解决了之前 PR #45096 中识别但未完全处理的竞态条件之一——分配时清零与已到达 KV 的冲突。

实现拆解

1. 新增 KVCacheManager 方法：在 vllm/v1/core/kv_cache_manager.py 中添加 get_zeroing_block_ids_in_range() 和 record_blocks_for_zeroing()。前者根据 token 范围返回将被远程KV加载覆盖的注意力块ID，后者用于在加载失败时重新记录需要清零的块。
2. 调度器排除覆盖块：在 vllm/v1/core/sched/scheduler.py 的 schedule() 中，当处理异步 KV 加载请求时，调用 get_zeroing_block_ids_in_range() 获取被覆盖块，存入 _skip_zero_block_ids 集合。修改 _get_new_block_ids_to_zero() 方法，从新块 ID 中移除这些跳过块，并清空集合。
3. 暴露单类型管理器属性：在 vllm/v1/core/single_type_kv_cache_manager.py 中添加 records_new_block_ids 属性，用于判断该管理器的块是否需要清零，从而区分注意力组（需清零）和 Mamba 状态组（不清零）。
4. 单元测试：在 tests/v1/kv_connector/unit/test_nixl_connector_hma.py 中新增三个测试：验证零化块 ID 只覆盖加载的注意力块、验证调度器正确过滤连接器加载的块、验证失败加载后重新记录未写入块。

关键文件：

- vllm/v1/core/kv_cache_manager.py（模块 缓存管理器；类别 source；类型 core-logic；符号 get_zeroing_block_ids_in_range, record_blocks_for_zeroing）：核心文件：新增 get_zeroing_block_ids_in_range 和 record_blocks_for_zeroing 方法，提供被加载块的 ID 查询和失败重清零能力。

- vllm/v1/core/sched/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 _get_new_block_ids_to_zero）：核心调度逻辑：新增 _skip_zero_block_ids 集合，在 schedule 中收集被加载覆盖的块，在 _get_new_block_ids_to_zero 中过滤，最后传给 worker。
- tests/v1/kv_connector/unit/test_nixl_connector_hma.py（模块 连接器测试；类别 test；类型 test-coverage；符号 _FakeBlock, _FakeSingleTypeManager, _make_fake_kv_cache_manager, test_zeroing_block_ids_cover_only_loaded_attention_blocks）：新增三个单元测试，验证零化块 ID 范围过滤、调度器跳过逻辑以及失败加载后重新记录，覆盖主要代码路径。
- 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 records_new_block_ids）：新增 records_new_block_ids 属性，用于区分注意力组（需要清零）和 Mamba 组（不需要），是判断的基础。

关键符号：get_zeroing_block_ids_in_range, record_blocks_for_zeroing, _get_new_block_ids_to_zero, records_new_block_ids

关键源码片段

vllm/v1/core/kv_cache_manager.py

核心文件：新增 get_zeroing_block_ids_in_range 和 record_blocks_for_zeroing 方法，提供被加载块的 ID 查询和失败重清零能力。

```
def get_zeroing_block_ids_in_range(
    self, request_id: str, start_token: int, end_token: int
) -> list[int]:
    """返回请求块中属于 [start_token, end_token) 范围的块 ID，
    仅包含那些被标记为需要清零的管理器（注意力组）。
    """
    ids: list[int] = []
    for mgr in self.coordinator.single_type_managers:
        if mgr.records_new_block_ids: # 仅注意力组需要清零
            start_idx = start_token // mgr.block_size
            end_idx = cdiv(end_token, mgr.block_size)
            blocks = mgr.req_to_blocks[request_id]
            ids.extend(blk.block_id for blk in blocks[start_idx:end_idx])
    return ids

def record_blocks_for_zeroing(self, request_id: str, start_token: int) -> None:
    """重新记录请求从 start_token 开始的块到清零列表，
    用于 KV 加载失败后的回退。start_token 必须块对齐。
    """
    for mgr in self.coordinator.single_type_managers:
        if mgr.records_new_block_ids:
            assert start_token % mgr.block_size == 0
            start_idx = start_token // mgr.block_size
            blocks = mgr.req_to_blocks[request_id]
            mgr.new_block_ids.extend(blk.block_id for blk in blocks[start_idx:])
```

vllm/v1/core/sched/scheduler.py

核心调度逻辑：新增 _skip_zero_block_ids 集合，在 schedule 中收集被加载覆盖的块，在 _get_new_block_ids_to_zero 中过滤，最后传给 worker。

```
def _get_new_block_ids_to_zero(self) -> list[int] | None:
    # 每步清空 new_block_ids 以防止无限增长
    new_block_ids_to_zero = self.kv_cache_manager.take_new_block_ids()
    if not self.needs_kv_cache_zeroing:
        return None

    if self._skip_zero_block_ids:
        # 移除将被远程 KV 加载覆盖的块
        skip = self._skip_zero_block_ids
        new_block_ids_to_zero = [b for b in new_block_ids_to_zero if b not in skip]
        skip.clear() # 清空集合，避免重复跳过

    return new_block_ids_to_zero or None # 空列表转为 None
```

tests/v1/kv_connector/unit/test_nixl_connector_hma.py

新增三个单元测试，验证零化块 ID 范围过滤、调度器跳过逻辑以及失败加载后重新记录，覆盖主要代码路径。

```
@pytest.mark.cpu_test
def test_zeroing_block_ids_cover_only_loaded_attention_blocks():
    """仅注意力组（records_new_block_ids=True）的块被返回，
    切片到给定 token 范围。Mamba 块不被清零。"""
    manager = _make_fake_kv_cache_manager()
    # 本地缓存 token [0,16)，远程加载覆盖 [16,56)
    assert manager.get_zeroing_block_ids_in_range("req-1", 16, 56) == [11, 12, 13]

@pytest.mark.cpu_test
def test_scheduler_filters_connector_loaded_blocks_from_zeroing():
    """在 _skip_zero_block_ids 中的块不会出现在零化列表中。"""
    scheduler = object.__new__(Scheduler)
    scheduler.needs_kv_cache_zeroing = True
    scheduler.kv_cache_manager = FakeKVCacheManager()
    scheduler._skip_zero_block_ids = {10, 12}
    assert scheduler._get_new_block_ids_to_zero() == [9, 11]
    assert not scheduler._skip_zero_block_ids # 应已清空
```

评论区精华

1. njhill 对连接器 API 方法的担忧：最初提交通过一个 KVConnectorMetadata.get_blocks_to_skip_kv_cache_zeroing 新方法实现，njhill 指出这会给每个加载连接器增加必须实现的方法，且隐式要求连接器清空未使用的尾部，存在遗漏的失败路径。
2. njhill 的替代方案：njhill 推送了一个额外提交，通过从 get_num_new_matched_tokens 返回的外部令牌授权直接推导需跳过的块，去除了连接器API方法，使逻辑集中在调度器侧。

3. 作者立场：arpera表示需要任何修复以发布可复现的结果，即使不覆盖所有连接器也接受。最终 njhill 的方案被合并。

- 连接器 API 方法 vs 调度器推导 (design): njhill 提交了替代方案，通过 get_num_new_matched_tokens 返回的 token 范围在调度器侧直接推导被覆盖块，移除了连接器 API 方法。
- 失败路径处理 (correctness): arpera 在后续提交中添加了 record_blocks_for_zeroing 方法，在失败路径中被调用，重新记录未写入块。

风险与影响

- 风险：
  - 回归风险：若 _skip_zero_block_ids 集合并未正确覆盖所有被加载块，可能导致部分块被错误跳过清零，引发精度问题。测试覆盖了正常和失败路径，但未覆盖极端并发场景。
  - 性能影响：每次异步加载时需遍历管理器列表计算块 ID，但范围较小，开销可忽略。
  - 连接器兼容性：当前方案不依赖连接器实现，但需连接器正确报告 num_new_local_computed_tokens 和 num_computed_tokens，否则可能错误跳过清零。
  - 错误处理：record_blocks_for_zeroing 在失败路径中被调用，但断言 start_token 块对齐，若传入未对齐值会导致崩溃。
- 影响：
  - 用户影响：使用混合模型（如 Qwen3.5、NemotronH）且启用 PD 异步调度的用户将恢复精度，无需回退到 --no-async-scheduling。
  - 系统影响：仅在 KV 连接器启用异步加载时激活，不影响非 PD 或同步调度场景。
  - 团队影响：维护了调度器与连接器之间的清晰接口，降低了后续连接器实现的耦合。
  - 风险标记：核心调度路径变更，依赖连接器正确报告 token 数，失败路径断言可能崩溃

关联脉络

- PR #47373 [KV Connector] Fix PD async scheduling for Qwen3.5 model: 本 PR 基于该 PR 的原始修复，进行整合和重新实现。
- PR #45096 [Mamba][PD] support async scheduling for mamba PD: 先前尝试修复同一问题，但识别出两个不同竞态条件，仅修复了其中一个。本 PR 解决了剩下的分配时清零竞态。
- PR #45357：PR #45096 讨论中提到的替代修复方案，用于解决第二个竞态（释放块重用）。
- PR #35219：引入分配时清零的功能，本 PR 解决的竞态正是其与 RDMA 传输的冲突。