PR #48102 完整报告

vllm-project/vllm

[Bugfix][KV Offloading] Fix stale transfer_jobs after reset_cache + harden job completion

合并时间：2026-07-13 01:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48102

执行摘要

- 一句话：修复 reset_cache 后 transfer_jobs 残留导致请求永久卡住
- 推荐动作：值得精读两个点：1）单个字典未清理如何导致请求永久卡死——展示了分布式系统中状态一致性的微妙陷阱；2）reviewer 要求精简 PR（去掉注释、去掉测试）——反映了对小型修复追求最小变更的原则。

功能与动机

PR body 明确描述了 bug 根因和复现步骤：reset_cache() 通过 _jobs.clear() 丢弃了所有进行中的任务，并通过 _stale_job_threshold 过滤过期 worker 响应，但未清除活跃请求的 transfer_jobs。由于陈旧 job_id 永远无法通过完成路径移除，导致请求永久卡在 step_skipped_waiting 状态。这是 KV Offloading 路线图（#33689）中 reset_cache 支持的一部分。

实现拆解

1. 定位修复点：在 vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py 的 reset_cache() 方法中，找到遍历 _req_status 并重置 group_state.next_stored_block_idx 的循环。
2. 增加单行修复：在循环体内部，对每个活跃的请求状态，在重置存储进度之后，添加 status.transfer_jobs.clear()，清除该请求残留的传输任务记录。
3. 修复位置验证：该操作在重置存储进度和丢弃全局任务（_jobs.clear()）之间，确保 transfer_jobs 与全局状态一致。
4. 无其他变更：未涉及测试、配置或部署配套改动，reviewer 明确要求精简 PR 范围。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 reset_cache）：唯一变更文件，在 reset_cache 方法中新增一行清除 transfer_jobs，修复因残留 job 导致请求永久卡住的 bug。

关键符号：reset_cache

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

唯一变更文件，在 reset_cache 方法中新增一行清除 transfer_jobs，修复因残留 job 导致请求永久卡住的 bug。

```
# vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

class KVConnectorOffloadingScheduler:
    # ... 其他方法 ...

    def reset_cache(self) -> None:
        """Reset the offloading manager cache, evicting all stored blocks."""

        # reset_cache cannot be called in the middle of a schedule step
        assert not self._current_batch_load_jobs
        assert not self._current_batch_jobs_to_flush
        assert not self._current_batch_allocated_block_ids

        # Flush all in-flight jobs
        self._current_batch_jobs_to_flush.update(self._jobs.keys())

        for req_id, status in list(self._req_status.items()):
            if status.req.is_finished():
                del self._req_status[req_id]

        # Reset offloading manager cache
        self.manager.reset_cache()

        # Reset store progress so active requests re-offload from block 0
        for status in self._req_status.values():
            for group_state in status.group_states:
                group_state.next_stored_block_idx = 0
            # --- 修复：清除残留的 transfer_jobs ---
            # 在 reset_cache 中，旧的 job_id 已被 _stale_job_threshold 废弃，
            # 如果不清理，这些 job_id 将永远无法被移除，
            # 导致 get_num_new_matched_tokens() 始终返回 None，请求永久卡死。
            status.transfer_jobs.clear()

        # Discard jobs and save job_counter to be able to discard worker responses
        self._stale_job_threshold = self._job_counter
        self._jobs.clear()
        self._block_id_to_pending_jobs.clear()

        # The manager pool is empty; pending event payloads and announced
        # reference counts are stale.
        self._events_tracker.reset()

        # Note: _current_batch_jobs_to_flush is intentionally NOT cleared.
        # The load flush IDs collected above must be delivered to workers.
        if self._blocks_being_loaded is not None:
            self._blocks_being_loaded.clear()
```

评论区精华

发现者 Alex-ai-future 提交了包含额外注释的版本。reviewer orozery 指出："Let's reduce the PR to just this line (no need for the comment above as well, and no need for new tests)." 随后提交者移除了注释，只保留核心一行。评审结论为 APPROVED。

- 精简 PR：只保留核心一行修复，移除注释和测试 (style): 提交者遵照执行，最终只保留一行 status.transfer_jobs.clear()。

风险与影响

- 风险：风险极低：仅新增一行 clear() 调用，且该调用在 reset_cache 场景下语义正确（所有正在进行的工作已被丢弃，缓存已被重置）。若存在某些地方依赖 reset_cache 后 transfer_jobs 残留来恢复状态，则可能被破坏，但从代码逻辑看，残留的 job_id 已不可达，清除是唯一正确的行为。
- 影响：直接影响 KV Offloading 功能中 reset_cache 路径的正确性。影响面窄，仅作用于 reset_cache 时处于活跃状态的请求。修复后这些请求可以正常继续调度，不再永久卡死。对非 offloading 场景无影响。
- 风险标记：单行修复，无测试覆盖

关联脉络

- PR #47987 Make tiering offload region DP-replica aware: 同为 KV Offloading 模块的 bugfix/refactor，修改了相关状态清理逻辑，关注同一功能线的稳定性。