PR #45823 完整报告

vllm-project/vllm

[Fix][KV offload] Defer `on_request_finished` until in-flight transfers drain

合并时间：2026-06-18 12:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45823

执行摘要

- 一句话：[KV offload] 延迟 on_request_finished 直至在途传输完成
- 推荐动作：此 PR 值得精读，特别是对 KV 卸载模块的正确性有直接影响。设计上采用的延迟回调模式（defer until drain）是一种常见的异步资源释放策略，值得参考。审核过程对 reset_cache 安全顺序的讨论也体现了小心处理重置时机的重要性。

功能与动机

根据 PR body，当前 submit_store() 可能在 SecondaryTierManager 的 on_request_finished() 之后被调用：连接器在 GPU→primary 存储仍在传输中时就急切地调用 on_request_finished()，后续传输完成才驱动 complete_store() → submit_store() 级联到 secondary tier。此 PR 修复该时序问题。关联 roadmap issue #33689。

实现拆解

1. 修改 update_connector_output（vllm/distributed/.../scheduler.py）：当作业完成且从 transfer_jobs 移除后，若请求已结束且 transfer_jobs 为空，则调用 self.manager.on_request_finished(req_status.req_context)，并删除 _req_status 条目。
2. 修改 request_finished：当被调度器调用时，若请求无在途作业，则立即调用 on_request_finished 并清理；若有在途作业，则延迟到 update_connector_output 处理，同时将非滑动窗口块注册到 _block_id_to_pending_jobs 防止块重用。
3. 修改 reset_cache：在调用 self.manager.reset_cache() 之前，遍历 _req_status，对已完成且有延迟调用的请求先调用 on_request_finished 并删除其状态，防止因作业被丢弃而泄漏。
4. 更新基类文档：在 vllm/v1/kv_offload/base.py 和 tiering/base.py 中明确 on_request_finished 的调用时机（请求无在途传输作业时触发，但已提交的异步传输可能仍在进行）。
5. 添加测试：新增 test_no_offload_call_after_on_request_finished 和 test_reset_cache_finalizes_finished_request_with_pending_store，验证延迟语义和 reset_cache 的更正。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 卸载调度器；类别 source；类型 core-logic；符号 OffloadingConnectorScheduler.update_connect

or_output, OffloadingConnectorScheduler.request_finished,OffloadingConnectorScheduler.reset_cache）：核心逻辑修改：实现on_request_finished 延迟调用和 reset_cache 泄漏修复。

- tests/v1/kv_connector/unit/offloading_connector/test_scheduler.py（模块 卸载调度器；类别 test；类型 test-coverage；符号 test_no_offload_call_after_on_request_finished, test_reset_cache_finalizes_finished_request_with_pending_store）：新增两个关键测试用例，确保延迟语义和 reset_cache 泄漏修复的正确性。
- vllm/v1/kv_offload/tiering/base.py（模块 分层卸载；类别 source；类型 documentation）：更新了 on_request_finished 的文档，明确调用时机和限制。
- vllm/v1/kv_offload/base.py（模块 卸载基类；类别 source；类型 documentation）：更新了 on_request_finished 的文档，明确调度器延迟调用且数据可能未持久化。

关键符号：OffloadingConnectorScheduler.update_connector_output, OffloadingConnectorScheduler.request_finished, OffloadingConnectorScheduler.reset_cache, test_no_offload_call_after_on_request_finished, test_reset_cache_finalizes_finished_request_with_pending_store

关键源码片段

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

核心逻辑修改：实现 on_request_finished 延迟调用和 reset_cache 泄漏修复。

```
# scheduler.py - update_connector_output 中的延迟调用
    if not req_status.transfer_jobs and req_status.req.is_finished():
        self.manager.on_request_finished(req_status.req_context)
        del self._req_status[job_status.req_id]

# scheduler.py - request_finished 延迟逻辑
    if not req_status.transfer_jobs:
        self.manager.on_request_finished(req_status.req_context)
        del self._req_status[request.request_id]
        return False, None
    # 注册 pending blocks...

# scheduler.py - reset_cache 中完成延迟请求
    for req_id, status in list(self._req_status.items()):
        if status.req.is_finished():
            self.manager.on_request_finished(status.req_context)
            del self._req_status[req_id]
    self.manager.reset_cache()
```

评论区精华

- 测试函数命名：作者最初用 _run() 简化测试，审核者 orozery 提议改用公共 run() 方法，因为 async/sync 已无区别。作者合并 main 后采纳。

- reset_cache 调用顺序：orozery 建议在 manager.reset_cache() 之前先处理延迟的 on_request_finished，以避免清理后丢失状态。作者按建议修改。
  - 测试使用 _run() vs run() (testing): 最终使用了 run()。
  - reset_cache 中 on_request_finished 的调用顺序 (correctness): 按建议修改：在 reset_cache 中先处理已完成请求的延迟调用，再调用 manager.reset_cache()。

风险与影响

- 风险：
  - 时序敏感：如果 update_connector_output 中判断 not req_status.transfer_jobs and req_status.req.is_finished() 的条件有误，可能导致 on_request_finished 被跳过或重复。但测试覆盖了正常路径。
  - reset_cache 变更：在 manager.reset_cache() 前调用 on_request_finished 可能触发额外的副作用，但该调用本身只应释放簿记，不应有 I/O 操作（按文档）。风险较低。
  - 影响范围：变更局限在 KV 卸载连接器的 scheduler 和基类 docstring，不影响其他子系统。
- 影响：
  - 用户：修复了次级 tier 存储在请求结束后仍意外调用的 bug，提高卸载可靠性和数据一致性。
  - 系统：reset_cache 不再泄漏请求状态，减少了内存使用（但影响微小）。
  - 团队：代码基语义更清晰，基类文档明确了 on_request_finished 的契约。
  - 风险标记：时序依赖，reset_cache 泄漏修复，核心调度路径变更

关联脉络

- PR #39726 [SimpleCPUOffloadConnector]: Add support for reset_cache(): 该 PR 在 SimpleCPUOffloadConnector 中添加了 reset_cache 支持，而本 PR 修复了 offloading scheduler 中 reset_cache 的泄漏，属于同一功能线的演进。
- PR #45679 [Test][KV Connector] Add request_finished fence population tests for offloading scheduler: 该 PR 为 offloading scheduler 添加了 fence 测试，本 PR 进一步增加延迟语义测试，两者共同提升测试覆盖。