PR #50926 完整报告

vllm-project/vllm

[CI][Bugfix] Fix flaky `test_store_orders_after_compute_write`

合并时间：2026-08-04 08:54

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50926

执行摘要

- 一句话：修复 simple_kv_offload 测试的跨阶段流竞争导致的偶发失败。
- 推荐动作：这是一个小而精准的 CI 稳定性修复，值得快速合入。它展示了一个有价值的调试思路：当自校验竞态测试出现“修复组反而失败”的反直觉现象时，要考虑前一阶段遗留的异步内核造成的跨阶段污染，而不是急着怀疑 barrier 逻辑本身。

功能与动机

该测试是一个自校验的竞态测试：无 barrier 的控制组必须实际发生竞态（control > 0），而带 barrier 的修复组必须完全干净（fixed == 0）。但在 CI 上出现了 store raced compute even with the barrier: 1 corrupt 的失败，说明 barrier 实验组被上一阶段（无 barrier 控制组）遗留的 in-flight 内核污染，导致测试结果不稳定，需要让每个阶段自包含。

实现拆解

1. 定位根因：_drive_store 的无 barrier 控制阶段中，store 只等待 store-copy 事件，而 compute stream 上的 sleep+fill 内核不受约束，host 循环会继续推进，留下约 800ms 的内核积压。
2. 修复方式：在 _drive_store 返回前（循环结束后）对 compute_stream 调用 synchronize()，确保该阶段所有计算内核执行完毕后再进入下一个阶段。
3. 不影响测试语义：该同步仅发生在每个阶段内部，不影响阶段内的竞态探测逻辑，且不改变两个断言的含义。

关键文件：

- tests/v1/simple_kv_offload/test_worker.py（模块 测试；类别 test；类型 test-coverage；符号 _drive_store, test_store_orders_after_compute_write）：这是本 PR 唯一修改的文件，在 _drive_store 返回前增加 compute stream 排空，消除跨阶段竞态导致的 flaky 失败。

关键符号：_drive_store, test_store_orders_after_compute_write

关键源码片段

tests/v1/simple_kv_offload/test_worker.py

这是本 PR 唯一修改的文件，在 _drive_store 返回前增加 compute stream 排空，消除跨阶段竞态导致的 flaky 失败。

```
# tests/v1/simple_kv_offload/test_worker.py
def _drive_store(backend, gpu, cpu, with_barrier: bool) -> int:
    compute_stream = torch.cuda.Stream()
    corrupt = 0
    for it in range(ITERS):
        val = (it % 126) + 1 # 1..126; distinct from the zero-initialized pool
        with torch.cuda.stream(compute_stream):
            torch.cuda._sleep(SLEEP_CYCLES)
            gpu.fill_(val)

        wait_event = None
        if with_barrier:
            wait_event = torch.Event()
            wait_event.record(compute_stream)

        store_events: list[tuple[int, torch.Event]] = []
        backend.launch_copy(
            block_ids,
            block_ids,
            is_store=True,
            event_idx=it,
            events_list=store_events,
            wait_event=wait_event,
        )

        deadline = time.time() + 10.0
        while not store_events and time.time() < deadline:
            time.sleep(0.0005)
        assert store_events, "background copy was never enqueued"
        store_events[0][1].synchronize()

        if int((cpu[:, 0].to(torch.int32) != val).sum().item()):
            corrupt += 1

    # 排空 compute stream 后再返回：无 barrier 控制阶段中 store 从不等待
    # compute，host 循环会领先很远，留下大量 sleep+fill 内核在飞。若不排空，
    # 上一阶段遗留的 fill 会与 barrier 阶段的 fill->copy 窗口在共享的 gpu
    # tensor 上竞争，偶发破坏一个迭代，导致 'store raced compute even with
    # the barrier' 断言误报。
    compute_stream.synchronize()
    return corrupt
```

评论区精华

没有实质性的评审讨论。两个 review 分别来自 claude[bot]（指出 fork 仓库自动评审被禁用，需维护者手动触发）和 tlrmchlsmth（直接 APPROVED）。PR 作者 njhill 在 commit

message 中详细解释了失败机制。

- fork 仓库自动 review 被禁用 (other): tlrmchlsmth 直接批准了该 PR，说明人工 review 完成。

风险与影响

- 风险：
  1. 该改动只影响测试代码，不影响任何生产逻辑，回归面极低。
  2. compute_stream.synchronize() 会让 _drive_store 阶段之间多一次主机端等待，可能略微增加测试耗时，但能稳定消除跨阶段竞态。
  3. 一个潜在风险是：如果未来该测试里的"无 barrier 必须竞态"的语义需要调整，新加的同步点可能会掩盖真正的问题，但这是测试代码自身的权衡，风险很小。 - 影响：影响范围限定于 tests/v1/simple_kv_offload/test_worker.py 这一个测试文件，面向 CI 稳定性，消除一个偶发失败点。对用户无影响，对团队而言减少了 CI 红盘与重试成本。 - 风险标记：仅测试代码变更，跨阶段异步竞态，CI 稳定性

关联脉络

- PR #48120 [Hybrid] Stage the postprocess inputs with a single loop over the request list: 同处 vllm/v1/worker 相关测试域，且同样涉及 mamba 后处理与测试稳定性，可作为 v1 worker 测试生态的参照。
- PR #50432 [Bugfix][Hybrid] Fix cross-block race on num_accepted in MRv2 align prefix cache: 同为 v1 worker 内核竞态类 bugfix，展示了该模块对异步竞态问题的持续关注。