PR #52144 完整报告

vllm-project/vllm

[Test] Add pause/resume E2E tests

合并时间：2026-08-18 14:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52144

执行摘要

- 一句话：新增 pause/resume 全链路 E2E 测试，双模 MRV1/MRV2 覆盖
- 推荐动作：值得精读。重点学习两处设计：一是用流式请求 + 线程事件同步生成进度来替代固定延时，解决“200-lie”下时序断言不稳定的难题；二是通过模块级 server fixture 配合 autouse 状态恢复，在 N 个用例与双 MRV 模式下复用服务进程，兼顾覆盖率与运行时间。由于该 PR 是纯测试补充，阅读时建议结合 RFC #45585 的矩阵规划与 #45586 的原方案对比，理解为何要把 pause/resume 从 sleep/wake 大套件中独立出来。

功能与动机

RFC #45585 指出 vLLM 作为 RL 主流 rollout 引擎，其 scheduler gating（pause/resume）和 weight-transfer 协议等 RL 专属代码面长期没有专用 CI 覆盖，且已出现多起线上 bug（如 #45326、#45518、#44972 等）。此前 #45586 曾将这些行为测试嵌套在 sleep/wake 生命周期套件中，依赖固定延时和宽泛的 liveness 断言，时序不稳定。本 PR 把 pause/resume 独立成模块，用流式请求同步真实的生成进度替代定时等待，强化可观察的 API 契约。

实现拆解

实现分三步：

1. 新增共享测试基础设施（conftest.py）：定义 server 上下文管理器，以子进程方式拉起带 VLLM_SERVER_DEV_MODE=1 的开发路由服务，轮询 /health 直到就绪；提供 poll_until 轮询助手以应对 sleep/wake/pause 接口的“200-lie”问题；定义 gen、ok、StreamResult、stream_completion、start_stream 等生成与流式 HTTP 助手；并封装 pause、resume、completion_with_cache_details、golden_output 等接口调用工具。
2. 编写 pause/resume 生命周期测试（test_pause_resume.py）：使用模块级 server_url fixture，通过 parameterize 的 use_v2 在 MRV1 与 MRV2 两种模型运行器下各启动一次服务；restore_unpaused_state autouse fixture 在每条用例前后强制 resume 以恢复状态。四个测试分别验证：多轮 pause/resume 的幂等性与 is_paused 状态；非法 mode 返回 400 且不影响原有状态；abort/wait/keep 三种模式下 in-flight 请求、暂停期间新请求及恢复后行为的差异；clear_cache 对 golden output 与 prefix cache 命中控制。
3. 目录结构配套：新增 state_transitions/__init__.py 空文件，将 RL 测试按路由分区（state_transitions、info_retrieval、consistency），与 RFC #45585 规划的目录布局对齐。

配套方面，测试依赖 VLLM_USE_V2_MODEL_RUNNER 环境变量切换模型运行器，并通过 --enable-prefix-caching 与 --enable-prompt-tokens-details 暴露 cache 与 tokens 明细，属于纯测试配置，不涉及 schema 或部署改动。

关键文件：

- tests/entrypoints/serve/dev/rlhf/conftest.py（模块 测试夹具；类别 test；类型 test-coverage；符号 server, poll_until, gen, gen_with_logprobs）：RL 生命周期测试套件的共享基础设施，提供 server 启动、轮询、流式生成、pause/resume HTTP 助手，是整套测试可运行与防 flaky 的关键。
- tests/entrypoints/serve/dev/rlhf/state_transitions/test_pause_resume.py（模块 生命周期测试；类别 test；类型 test-coverage；符号 use_v2, server_url, restore_unpaused_state, TestPauseResume）：核心测试文件，定义 MRV1/MRV2 双模 fixture 与四个 P0/P1 用例，直接验证 pause/resume 的 API 契约与请求生命周期。
- tests/entrypoints/serve/dev/rlhf/state_transitions/__init__.py（模块 测试目录；类别 test；类型 test-coverage）：空包标记文件，将 tests/entrypoints/serve/dev/rlhf 下按 RFC 规划的测试子目录正式成为 Python 包。

关键符号：server, poll_until, start_stream, stream_completion, pause, resume, golden_output, test_state_and_idempotency_across_cycles, test_invalid_mode_preserves_state, test_mode_request_lifecycle, test_clear_cache_preserves_output_and_controls_prefix_cache

关键源码片段

tests/entrypoints/serve/dev/rlhf/conftest.py

RL 生命周期测试套件的共享基础设施，提供 server 启动、轮询、流式生成、pause/resume HTTP 助手，是整套测试可运行与防 flaky 的关键。

```
def poll_until(
    predicate: Callable[[], bool],
    timeout: float = 10.0,
    interval: float = 0.5,
) -> bool:
    """轮询 predicate 直到为 True 或超时。

    针对 vLLM sleep/wake 的 200-lie 现象:
    HTTP 接口可能在底层操作完成前就返回 200,
    所以需要此助手来验证操作后的真实状态，而不是信任 200 即完成。
    """
    deadline = time.time() + timeout
    while time.time() < deadline:
        try:
            if predicate():
                return True
        except Exception:
            pass # 忽略瞬时错误，继续轮询
        time.sleep(interval)
```

```
    return False


def start_stream(url: str, max_tokens: int) -> tuple[StreamResult, threading.Thread]:
    """启动一个流式生成请求，并等待其真正产出首个 token。

    这里用 started 事件同步生成进度，而不是固定 sleep，
    从而避免慢机器上因延时不足导致请求尚未开始生成就暂停。
    若请求在 pause 前就已结束或启动失败，先 abort 再 resume 清理现场。
    """
    result = StreamResult()
    thread = threading.Thread(
        target=stream_completion,
        args=(url, result, max_tokens),
    )
    thread.start()
    started = result.started.wait(timeout=10)
    if not started or result.done.is_set():
        # 清理异常状态：先终止未完成请求，再恢复引擎
        pause(url, mode="abort")
        resume(url)
        thread.join(timeout=10)
    assert started, "request did not start generating"
    assert not result.done.is_set(), "request completed before it could be paused"
    return result, thread
```

tests/entrypoints/serve/dev/rlhf/state_transitions/test_pause_resume.py

核心测试文件，定义 MRV1/MRV2 双模 fixture 与四个 P0/P1 用例，直接验证 pause/resume 的 API 契约与请求生命周期。

```
@pytest.mark.parametrize(
    ("mode", "max_tokens", "inflight_finish_reason"),
    [
        pytest.param("abort", 256, "abort", id="abort"),
        pytest.param("wait", 256, "length", id="wait"),
        pytest.param("keep", 256, "length", id="keep"),
    ],
)
def test_mode_request_lifecycle(
    self,
    server_url,
    mode,
    max_tokens,
    inflight_finish_reason,
):
    # 先启动一个 in-flight 流式请求，并等它产出文本再 pause
    inflight, inflight_thread = start_stream(server_url, max_tokens)
    new_result: dict[str, Any] = {}
    new_done = threading.Event()
```

```
    def _new_request():
        # 在 pause 期间尝试提交的新请求，记录结果并置事件
        new_result["response"] = gen(server_url, max_tokens=4, timeout=60)
        new_done.set()

    new_thread = threading.Thread(target=_new_request)
    try:
        assert pause(server_url, mode=mode) == 200
        assert is_paused(server_url)

        if mode in ("abort", "wait"):
            # abort 和 wait 都会让 in-flight 请求立即结束
            assert inflight.done.is_set()
        else:
            # keep 模式冻结生成进度，chunk 数量不应继续增长
            chunks_after_pause = len(inflight.chunks)
            assert not inflight.done.wait(timeout=5)
            assert len(inflight.chunks) == chunks_after_pause, (
                "in-flight request continued generating in keep mode"
            )

        # 暂停期间提交的新请求必须保持 pending
        new_thread.start()
        assert not new_done.wait(timeout=0.3), (
            "new request completed while generation was paused"
        )
    finally:
        # 无论断言是否失败都先恢复，避免污染后续用例
        assert resume(server_url) == 200
        inflight_thread.join(timeout=30)
        if new_thread.ident is not None:
            new_thread.join(timeout=30)

    # 恢复后：旧请求按模式以预期 finish_reason 收尾，新请求正常完成
    assert not inflight_thread.is_alive()
    assert inflight.error is None
    assert inflight.finish_reason == inflight_finish_reason
    assert not new_thread.is_alive()
    assert ok(new_result.get("response"))
```

评论区精华

核心讨论集中在三处：

- 结构建议（Ronald1995）：review 评论指出 it's better to move these method to conftest.py，认为流式生成、线程同步等辅助方法应下沉到共享 conftest，随后 floatlibai 回复 done 并在最终提交中完成重构，形成现在的 conftest.py + test_pause_resume.py 分层。

- CI 接入确认（aoshen02）：aoshen02 询问是否需要加到 Buildkite workflow，floatlibai 说明 .buildkite/test_areas/entrypoints.yaml 中的 entrypoints-integration-api-server 已覆盖 tests/entrypoints/serve 目录，无需额外配置，aoshen02 表示认同并触发 /ci run。
- 后续行为契约缺口（floatlibai 自述）：floatlibai 在评论中主动指出 pause 后提交的新请求在 abort 模式下的期望行为需要更新，并提到 #51476 是后续两周内计划实现的 RFC，两个议题相关联，属于未完全闭合的待办。
  - 把流式与生命周期辅助方法下沉到 conftest.py (design): floatlibai 回复 done，并在最后一次 commit 中完成 conftest 重构与 import 调整。
  - 是否需要显式接入 Buildkite (question): floatlibai 说明 entrypoints-integration-api-server 已按目录 glob 覆盖 tests/entrypoints/serve，无需额外改动；aoshen02 表示 I see 并 LGTM。
  - abort 模式下暂停后新请求的期望行为需要更新 (question): 未在 PR 内落地，属于留待 #51476 推进时处理的后续事项。

风险与影响

- 风险：主要风险集中在测试本身：
- 时序敏感性：start_stream 依赖线程事件等待生成首个 token，test_mode_request_lifecycle 用 0.3 秒断言新请求在暂停期间不完成，在慢 CI 或高负载机器上可能产生 flaky。
- 资源占用与端口冲突：server 默认绑定固定端口 8770，若多个测试模块并行可能冲突；每次启动服务约 37 秒，整套测试约 110 秒，会拉长 entrypoints 相关 CI 时长。
- 外部依赖：默认使用 Qwen/Qwen3-0.6B 真实权重，离线或模型缓存缺失环境下 setup 会失败（有 dummy_weights 选项但当前用例未启用）。
- 契约变更风险：评论中提及的 abort 模式下暂停后新请求行为尚未更新断言，若实际服务行为与预期不一致，可能掩盖真实 bug 或导致用例过时。
  - 无生产代码改动，不存在对推理路径、性能或安全的直接影响。
- 影响：影响范围仅限测试目录 tests/entrypoints/serve/dev/rlhf/，对 vLLM 生产代码零影响。对团队而言，该 PR 补上了 RFC #45585 中 RL CI 矩阵的 P0/P1 关键空白，为 sleep/pause 相关的回归（如请求绕过调度器挂起、重复 pause 幂等性、clear_cache 语义）提供自动化防护。对下游 RL 框架（verl、OpenRLHF 等）而言，pause/resume 行为契约得到更明确的验证保障。MRV1/MRV2 双覆盖也让模型运行器迁移期的行为差异可被持续观测。
- 风险标记：测试时序敏感可能 flaky，固定端口 8770 存在并行冲突风险，依赖真实模型权重下载，abort 契约断言待随新 RFC 更新，CI 时长增加约 110 秒

关联脉络

- PR #45585 [RFC] RL CI Matrix for vLLM: Behavioral + Physical + Protocol Coverage: 本 PR 正是该 RFC 规划的 RL CI 矩阵中 scheduler gating（pause/resume）覆盖的一部分，PR body 开头明确指出 Purpose 来自 #45585。

- PR #45586 RL lifecycle test suite（最初实现 pause/resume 覆盖的 PR）：Ronald1995 在 review 中说明本 PR 把原先嵌在 #45586 大生命周期套件中的 TestPauseResume 独立出来并强化断言，conftest 中也引用了 #45586 链接。
- PR #51476 RL 相关新 RFC（评论中提及，待 2 周内实现）：aoshen02 在评论区提醒该 RFC 即将实现，floatlibai 据此指出 abort 测试行为需跟进，两个 PR 在暂停后请求语义上存在交集。