PR #51050 完整报告

vllm-project/vllm

[CI][Bugfix] Fix `test_shutdown_on_engine_failure` startup deadlock

合并时间：2026-08-05 09:29

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51050

执行摘要

- 一句话：修复关闭测试启动死锁：服务输出改文件重定向并移除 ROCm 特例
- 推荐动作：建议快速浏览而非精读：这是一个教科书级的 subprocess 管道缓冲区死锁修复，适合作为集成测试中规避子进程输出阻塞的参考模式——轮询就绪阶段用文件重定向、失败时再回读日志。主要供 CI/ 测试维护者学习；无需进入核心逻辑 review 清单。

功能与动机

PR body 明确指出根因：测试轮询服务器就绪时从不排空服务器的 stdout/stderr 管道，启动输出超过管道缓冲区后服务器阻塞在 write 上、永远无法就绪，最终报 'Server failed to start in 120 seconds'。修复方案是把输出重定向到文件，同时移除 ROCm 平台为躲避同样挂起而禁用管道捕获的特例，并让两类失败路径都能报告服务器输出。

实现拆解

变更全部集中在 tests/entrypoints/openai/completion/test_shutdown.py，分四步完成：

1. 调整导入与测试签名：删除对 vllm.platforms.current_platform 的依赖及 _IS_ROCM 常量，新增 from pathlib import Path；test_shutdown_on_engine_failure 的签名增加 tmp_path: Path fixture，为文件重定向提供临时目录。
2. 子进程输出从管道改为文件重定向：用 with server_log.open("wb") as log_file 包裹 subprocess.Popen，stdout=log_file、stderr=subprocess.STDOUT，同时移除原 text 参数和按平台条件选择管道的逻辑。由于日志直接写入文件，轮询就绪期间无需任何一方排空管道，从根上消除写阻塞死锁。
3. 统一失败诊断路径：删除 _IS_ROCM 分支，进程提前退出与启动超时两个失败路径统一在 pytest.fail 中通过 server_log.read_text(errors='replace') 附带服务端完整日志，提升 CI 定位效率。其余逻辑（就绪后 terminate 模拟崩溃、断言 API 连接错误）保持不变。
4. 配套验证：本次仅修改测试文件，无源码、配置或部署配套改动；tlrmchlsmth 直接 approve，AMD CI 负责人 AndreasKaratzas 确认该修复针对的管道死锁与 AMD CI 中观察到的另一失败是不同问题。

关键文件：

- tests/entrypoints/openai/completion/test_shutdown.py（模块 关闭流程；类别 test；类型 test-stability；符号 test_shutdown_on_engine_failure）：唯一变更文件，包含 test_shutdown_on_engine_failure 的死锁修复：子进程输出从管道改为文件重定向，移除

ROCm 特例并统一失败日志输出。

关键符号：test_shutdown_on_engine_failure

关键源码片段

tests/entrypoints/openai/completion/test_shutdown.py

唯一变更文件，包含 test_shutdown_on_engine_failure 的死锁修复：子进程输出从管道改为文件重定向，移除 ROCm 特例并统一失败日志输出。

```
# tests/entrypoints/openai/completion/test_shutdown.py
# 引擎崩溃时验证 API 返回连接错误的集成测试。
# 核心修复：子进程输出改为写入文件而非管道，
# 避免启动日志超过管道缓冲区时子进程阻塞在 write() 上、永不就绪。

@pytest.mark.asyncio
async def test_shutdown_on_engine_failure(tmp_path: Path):
    """Verify that API returns connection error when server process is killed."""

    port = get_open_port()

    # 轮询就绪期间无人排空管道，输出超过缓冲区（约 64 KiB）
    # 时子进程会写阻塞。改为文件重定向后该问题自然消失。
    server_log = tmp_path / "server.log"
    with server_log.open("wb") as log_file:
        proc = subprocess.Popen(
            [
                sys.executable,
                "-m",
                "vllm.entrypoints.openai.api_server",
                "--model", MODEL_NAME,
                "--dtype", "bfloat16",
                "--max-model-len", "128",
                "--enforce-eager",
                "--port", str(port),
                "--gpu-memory-utilization", "0.05",
                "--max-num-seqs", "2",
            ],
            # stdout 指向日志文件，stderr 合并进同一文件，
            # 同时移除原 ROCm 下禁用管道捕获的特例分支。
            stdout=log_file,
            stderr=subprocess.STDOUT,
            # 让子进程忽略 SIGINT，避免并发信号干扰测试进程组。
            preexec_fn=lambda: signal.signal(signal.SIGINT, signal.SIG_IGN),
        )

    # 轮询等待服务就绪
    start_time = time.time()
    client = openai.AsyncOpenAI(
```

```
        base_url=f"http://localhost:{port}/v1",
        api_key="dummy",
        max_retries=0,
        timeout=10,
    )

    while time.time() - start_time < _SERVER_STARTUP_TIMEOUT:
        try:
            await client.completions.create(
                model=MODEL_NAME, prompt="Hello", max_tokens=1
            )
            break
        except Exception:
            time.sleep(0.5)
            if proc.poll() is not None:
                # 进程提前退出：回读日志文件给出诊断信息
                pytest.fail(
                    f"Server died during startup: {proc.returncode}\n"
                    f"{server_log.read_text(errors='replace')}"
                )
    else:
        # 启动超时：终止进程并附带完整日志，便于定位卡点
        proc.terminate()
        proc.wait(timeout=_PROCESS_EXIT_TIMEOUT)
        pytest.fail(
            f"Server failed to start in {_SERVER_STARTUP_TIMEOUT} seconds\n"
            f"{server_log.read_text(errors='replace')}"
        )

    # 服务就绪后模拟崩溃：终止进程，确认后续 API 调用报连接错误
    proc.terminate()
    time.sleep(1)

    with pytest.raises((openai.APIConnectionError, openai.APIStatusError)):
        await client.completions.create(
            model=MODEL_NAME, prompt="This should fail", max_tokens=1
        )

    return_code = proc.wait(timeout=_PROCESS_EXIT_TIMEOUT)
    assert return_code is not None
```

评论区精华

无常规 review 评论，tlrmchlsmth 直接 approve；claude[bot] 因 fork PR 自动 review 被禁用。唯一实质讨论来自 AndreasKaratzas 的关联 Issue 评论：

> 我想看看这个修复是否也能解决这个测试组的问题…… 看起来那是另一个不同的 issue。

说明该修复目标明确、范围收敛，AMD CI 上其余失败不在本 PR 覆盖范围内。

- 修复是否同时覆盖 AMD CI 的另一个测试组失败 (question): 评论者随后更新确认那是另一个不同的 issue，不在本 PR 修复范围内。

风险与影响

- 风险：
  - 平台行为变更：ROCm 从原先的“不捕获输出”切到“文件重定向”，行为统一后需在 AMD CI 上回归确认；关联 Issue 评论表明 AMD CI 现有失败独立于本修复，因此被该 PR 引入新问题的概率很低。
  - 诊断差异：stdout 与 stderr 合并写入同一文件，日志顺序可能交错，可读性略降；read_text(errors='replace') 保证编码异常不会导致二次失败。
  - 环境依赖：依赖 pytest 的 tmp_path 提供磁盘写入，若 runner 文件系统异常可能导致启动失败，概率低。
  - 范围控制：仅测试文件变更，不存在对推理路径、API 行为或配置解析的回归风险。
  - 影响：影响范围限定在 CI 测试层：消除 120 秒假超时、减少 flaky 失败、统一平台行为并降低维护成本；失败诊断从仅返回码提升为完整服务日志。对 vLLM 运行时、API 与模型推理均无影响；对团队的直接收益是更稳定的 CI 信号和更快的失败定位。影响程度中等偏低。
  - 风险标记：测试逻辑变更，移除 ROCm 特例，无源码主路径影响

关联脉络

- PR #51068 Prune redundant tests points in correctness_e2e/[test_sequence_parallel,test_async_tp]: 同属近期 CI 测试矩阵精简与稳定性收敛工作线，与本次修复共享减少 CI 假失败的动机，但文件不重叠。
- PR #51069 [CI] Prune PyTorch Compilation Unit Tests: 同期 CI 稳定性维护工作，与本次修复同属 CI 基础设施改进脉络。