PR #47873 完整报告

vllm-project/vllm

[Rust Frontend] Fix flaky `tls_handshake_timeout_drops_silent_client` test

合并时间：2026-07-15 19:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47873

执行摘要

该 PR 修复了 Rust 前端 tls_handshake_timeout_drops_silent_client 测试的竞态条件。通过移除手动时钟推进，改为依赖 tokio 暂停时钟自然推进，同时新增 nextest 慢超时配置，确保测试在 CI 中稳定运行。

功能与动机

问题：测试手动推进暂停时钟，导致握手关闭与 socket read 之间存在竞态，测试不稳定。PR body 原文：'the test manually advanced the paused clock and raced the handshake close against the socket read, making it flaky.'

实现拆解

1. 简化测试逻辑：在 rust/src/server/src/tls_tests.rs 中，删除了 yield_now() 和 advance() 调用，连接后直接执行 tcp.read(&mut buf).await，让 tokio 暂停时钟在 read 等待期间自动推进到超时。
2. 保留断言语义：将断言从 Ok(Ok(0)) | Ok(Err(_)) 调整为 Ok(0) | Err(_)，适配去掉外层 timeout 后的结果类型。
3. 添加关闭时机验证：新增 start.elapsed() >= TLS_HANDSHAKE_TIMEOUT 断言，确保连接关闭确实发生在握手超时后。
4. 引入 nextest 慢超时：在 rust/.config/nextest.toml 新增配置，slow-timeout = { period = "60s", terminate-after = 2 }，为整个 Rust 工作区提供 120 秒的测试超时兜底。

rust/src/server/src/tls_tests.rs

测试修复的核心文件，修改了手握手超时测试的逻辑。

```
#[tokio::test(start_paused = true)]
async fn tls_handshake_timeout_drops_silent_client() {
    // Silent client (no ClientHello) must be dropped at the handshake deadline.
    let certs = TestCerts::generate();
    let (addr, shutdown) = spawn_server(Some(server_tls(&certs, 0))).await;

    let mut tcp = TcpStream::connect(&addr).await.expect("connect");
    let start = tokio::time::Instant::now();
    let mut buf = [0u8; 1];
    // 不再手动 advance 时钟，让 tokio 暂停时钟在 read 等待时自动推进
    let read = tcp.read(&mut buf).await;
```

```
    assert!(
        matches!(read, Ok(0) | Err(_)),
        "server must drop a stalled TLS handshake (expected close, got {read:?})"
    );
    // 验证关闭发生在握手超时之后，而非提前关闭
    assert!(
        start.elapsed() >= tls::TLS_HANDSHAKE_TIMEOUT,
        "closed too early to be the handshake deadline: {:?}",
        start.elapsed()
    );
    shutdown.cancel();
}
```

评论区精华

> Codex bot: 如果 TLS 握手超时退化，这个 bare read().await 没有测试本地超时，在非 nextest 环境下会无限等待。建议用超时包裹。 tahsintunan: 已通过 nextest 的 slow-timeout 配置覆盖，且 nextest 是 CI 的标准运行器。

最终未采纳超时建议，但达成的共识是 nextest 慢超时是足够的防御措施。

风险与影响

- 低风险：仅修改测试，不涉及生产代码。
- 影响范围：仅 Rust 前端测试套件，修复了已知不稳定测试，新增的 nextest 配置提高了整个 Rust workspace 的测试健壮性。
- 潜在风险：如果 tokio 暂停时钟行为在版本升级后变化，可能重入不稳定；但 CI 定期运行可及早发现。

关联脉络

该 PR 是独立的测试稳定性修复，不与任何历史 PR 直接关联。它展示了 Rust 测试中利用 tokio 暂停时钟的最佳实践，值得同一 workspace 的其他测试参考。