

PR #51557 完整报告

vllm-project/vllm

[CI] Stabilize DP supervisor lifecycle tests

合并时间: 2026-08-10 21:13

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51557>

执行摘要

- 一句话: 修复 DP supervisor 测试静默超时与连接重置误判
- 推荐动作: 建议快速浏览而非精读。这个 PR 是测试稳定性修复的典型范例, 值得借鉴的设计点: 用单调时钟截止时间替代固定重试次数、把静默返回改为显式抛错、按预期异常层级收窄而非断言具体子类。如果团队经常在 CI 上处理 flaky 测试, 可以参考其诊断与修复思路。另外该 PR 也展示了 AI 辅助 (OpenAI Codex 起草 + 人工逐行审查) 的协作流程。

功能与动机

PR body 说明, 在 CI 负载较高时 readiness helper 会静默耗尽 10 秒等待 (原 `_poll_until_api_server_running` 重试耗尽后静默返回, 调用方无法区分成功与超时), 而故意 kill 子进程产生的连接重置会被旧的异常断言拒绝 (只接受 `aiohttp.ClientConnectorError`, 实际可能抛 `ClientOSError` 等其他子类)。Main build 83054 上 `test_basic_lifecycle` 与 `test_failed_startup` 失败 (2 failed, 253 passed)。作者还指出 #50916 只改了 `uvicorn` signal handling, 没有 PR 修复这些测试 helper, 故需要本 PR 补齐。

实现拆解

1. 重写 readiness 轮询: 将 `tests/entrypoints/openai/test_dp_supervisor.py` 中的 `_poll_until_api_server_running` 从 `retries=10` 固定次数改为 `timeout_s=30.0` 的截止时间模型。函数以 `time.monotonic()` 计算 deadline, 循环内为每次请求设置 `aiohttp.ClientTimeout(total=min(2.0, remaining_s))`, 避免单次请求长时间挂起; 只要子进程 accept 请求 (返回任何状态码, 包括 503) 即视为就绪并返回。循环耗尽后抛出携带 `last_exc` 的 `TimeoutError`, 把静默失败变成显式失败, 便于定位。
2. 放宽 kill 路径异常: `_kill_server` 原来的 `except Exception as e: assert isinstance(e, aiohttp.ClientConnectorError)` 改为 `except aiohttp.ClientConnectionError: return`。由于 `/kill` 在请求处理器内部发送 `SIGKILL`, 连接必然被重置, `aiohttp` 可能抛出 `ClientConnectorError`、`ClientOSError` 等不同 `ClientConnectionError` 子类; 捕获基类既接受这些预期结果, 又保留其他异常让测试失败, 防止误吞真实错误。
3. 采纳评审建议: 根据 yewentao256 的 suggestion, 轮询的异常捕获从裸 `TimeoutError` 改为 `asyncio.TimeoutError` (head 版本已应用), 避免误捕获非 `asyncio` 场景的超时。
4. 测试与配套: 仅修改这一个测试文件, 无产品代码、配置、schema 或部署配套变更。作者本地执行 `pytest tests/entrypoints/openai/test_dp_supervisor.py -q` 验证, 分支上 16 个测试全部通过; CI 由 maintainer 触发 Buildkite 验证, 无新增测试成本。

关键文件：

- tests/entrypoints/openai/test_dp_supervisor.py（模块 DP 监督；类别 test；类型 test-coverage；符号 _poll_until_api_server_running, _kill_server）：唯一变更文件，集中了 readiness 轮询与 kill 异常处理两处稳定性修复。

关键符号：_poll_until_api_server_running, _kill_server

关键源码片段

tests/entrypoints/openai/test_dp_supervisor.py

唯一变更文件，集中了 readiness 轮询与 kill 异常处理两处稳定性修复。

```
async def _poll_until_api_server_running(
    port: int, timeout_s: float = 30.0, use_ssl: bool = False
) -> None:
    """Return once the child accepts a request; it reports 503 until healthy."""
    # 用 deadline 替代固定 retries，保证总等待时间可控，避免高负载下静默耗尽
    scheme = "https" if use_ssl else "http"
    url = f"{scheme}://127.0.0.1:{port}/health"
    deadline = time.monotonic() + timeout_s
    last_exc: Exception | None = None
    async with aiohttp.ClientSession() as session:
        while (remaining_s := deadline - time.monotonic()) > 0:
            try:
                # 单次请求带上限超时，不独占整个剩余窗口
                request_timeout = aiohttp.ClientTimeout(total=min(2.0, remaining_s))
                async with session.get(
                    url,
                    ssl=False if use_ssl else None,
                    timeout=request_timeout,
                ):
                    # 子进程 accept 请求即视为就绪；503 表示尚未 healthy 也算启动成功
                    return
            except (aiohttp.ClientError, asyncio.TimeoutError) as exc:
                last_exc = exc
                print("Test detected not started yet, sleeping for 1s")
                remaining_s = deadline - time.monotonic()
                if remaining_s > 0:
                    await asyncio.sleep(min(1.0, remaining_s))
    # 显式抛错代替静默返回，让测试失败原因一目了然
    raise TimeoutError(
        f"API server on port {port} did not start within {timeout_s}s"
    ) from last_exc
```

```
async def _kill_server(port: int, use_ssl: bool = False) -> None:
    # /kill 在请求处理内部发送 SIGKILL，成功 kill 无法返回 HTTP 响应，
    # 客户端只会看到连接被重置，因此接受所有连接级异常作为预期结果
    scheme = "https" if use_ssl else "http"
```

```
url = f"{scheme}://127.0.0.1:{port}/kill"
try:
    async with (
        aiohttp.ClientSession() as session,
        session.get(url, ssl=False if use_ssl else None) as resp,
    ):
        assert resp.status != 200
except aiohttp.ClientConnectionError:
    return
```

评论区精华

核心讨论集中在两处：

- `_kill_server` 是否刻意直接返回错误：yewentao256 在 line 544 提问 "Is this intentional? We directly return the error"。作者解释：/kill 在请求处理器内部发送 SIGKILL，成功的 kill 无法返回 HTTP 响应，客户端只会看到连接关闭；aiohttp 可能抛出 `ClientConnectorError`、`ClientOSError` 或其他 `ClientConnectionError`，捕获基类是为了接受这些预期的连接级结果，同时仍允许无关错误让测试失败。reviewer 认可后 APPROVED (LGTM)。
- 轮询异常类型建议：yewentao256 建议改为 `except (aiohttp.ClientError, asyncio.TimeoutError) as exc`，明确区分 `asyncio` 超时来源，该建议已被采纳进最终版本。
 - `_kill_server` 直接返回连接异常是否合理 (design)：作者确认有意为之，reviewer 接受并 APPROVED。
 - 轮询超时异常类型应明确为 `asyncio.TimeoutError (correctness)`：建议被采纳，head 版本已使用 `asyncio.TimeoutError`。

风险与影响

- 风险：虽为纯测试改动，仍有几点需留意：
 - 超时窗口：30.0 秒是硬编码默认值，且任何一次成功响应都会立即返回。在极端 CI 负载下如果子进程 30 秒内无法 `accept` 请求，测试仍会失败；好在新实现会抛出带 `last_exc` 的 `TimeoutError`，诊断信息比原来清晰。相关调用方如需更长等待可传入 `timeout_s`。
 - 异常放宽的轻微掩盖风险：`_kill_server` 现在把所有 `ClientConnectionError` 都视为预期结果，若连接因端口占用、目标错误等非预期原因失败，也会被当成成功返回。鉴于端口与 URL 在测试中固定，该风险很低。
 - 状态码不再校验：新轮询收到任何响应即返回，若子进程意外返回 500 等错误状态，测试会继续走到后续断言，错误可能表现为后续步骤失败而非 `readiness` 失败，定位时需注意。整体无产品运行时与安全风险。
- 影响：影响范围仅限 `tests/entrypoints/openai/test_dp_supervisor.py` 下的 DP supervisor 生命周期测试（约 16 个用例）。对最终用户、服务端推理路径零影响。对团队的价值在于消除了 CI 上的偶发假阳性（main 上 2 failed, 253 passed → 分支 16 passed），减少维护者反复重跑与排查成本；同时显式 `TimeoutError` 让未来类似失败更容易诊断。CI 成本不变，因为只修改了已有的测试用例。

- 风险标记：测试超时可能不足，异常捕获放宽，无产品代码变更

关联脉络

- PR #50916 [Frontend] Disable uvicorn signal handlers instead of racing them: PR body 明确说明 #50916 只改了 signal handling，本 PR 是其未覆盖的测试 helper 的后续修复；同属 DPSupervisor 生命周期稳定性工作线。