PR #25809 完整报告

sgl-project/sglang

deflake priority below-threshold test

合并时间：2026-05-20 06:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25809

执行摘要

- 一句话：修复优先级调度测试的竞态不稳定性
- 推荐动作：该 PR 值得快速合入，没有风险。可从中学习到处理异步测试 flaky 的通用模式：用 create_task + 显式延迟代替 gather 来控制请求到达顺序；使用 ignore_eos 确保请求持续运行，避免因提前结束导致的断言失败。

功能与动机

测试用例 test_priority_scheduling_preemption_below_threshold_validation 在 CI 中不稳定（7/10 通过率）。根本原因是 asyncio.gather 并发发送请求时，两个请求的到达顺序不确定，当高优先级请求先到达时，低优先级请求的 e2e_latency 可能更短，导致断言失败。

实现拆解

变更仅涉及 test/registered/scheduler/test_priority_scheduling.py 文件，通过三步重构解决 flaky 问题：

1. 引入分批发送机制：将原来的单次并发发送拆解为两个独立的 asyncio.create_task，通过 await asyncio.sleep(1.0) 确保优先级为 0 的请求先被服务器处理并进入运行队列，然后优先级为 5 的请求才到达，模拟低优先级请求被抢占的场景。
2. 调整参数缩短测试时间：将 max_new_tokens 从 10000 减少到 1000，并添加 ignore_eos=True，使请求在生成足够 token 前不会因 EOS 令牌提前结束，保证优先级为 0 的请求在整个测试期间持续运行，同时大幅减少测试执行时间，降低因超时引入的噪声。
3. 重构辅助函数：提取 _send 和 _run 局部辅助函数，使逻辑更清晰，同时将请求构造扁平化，简化代码。

关键文件：

- test/registered/scheduler/test_priority_scheduling.py（模块 测试；类别 test；类型 test-coverage；符号 _send, _run）：唯一变更文件，修复了优先级调度抢占验证测试的不稳定性

关键符号：_send, _run

关键源码片段

test/registered/scheduler/test_priority_scheduling.py

唯一变更文件，修复了优先级调度抢占验证测试的不稳定性

```
# 修复后的 test_priority_scheduling_preemption_below_threshold_validation 函数
# 核心思想：通过分批发送控制请求到达顺序，避免 asyncio.gather 的竞态

def test_priority_scheduling_preemption_below_threshold_validation(self):
    """Verify running requests are not preempted by requests with priorities below preemption
    threshold"""

    # Stagger sends so priority=0 occupies the running queue before
    # priority=5 arrives -- asyncio.gather gives no arrival-order guarantee.
    # ignore_eos on both: priority=0 stays running when priority=5 arrives
    # (exercises the no-preempt path), and priority=5's runtime must exceed
    # the stagger so its server-side e2e_latency stays > priority=0's.
    async def _send(priority, **sampling):
        return await send_concurrent_generate_requests_with_custom_params(
            self.base_url,
            [{"priority": priority, "sampling_params": sampling}],
        )

    async def _run():
        # 先发送优先级为 0 的请求，等待 1 秒确保其进入运行队列
        first = asyncio.create_task(_send(0, max_new_tokens=1000, ignore_eos=True))
        await asyncio.sleep(1.0)
        # 然后发送优先级为 5 的请求，此时优先级 0 的请求应仍在运行
        second = asyncio.create_task(_send(5, max_new_tokens=1000, ignore_eos=True))
        return (await first) + (await second)

    responses = asyncio.run(_run())

    expected_status_and_error_messages = [
        (200, None),
        (200, None),
    ]

    e2e_latencies = []
    _verify_genereate_responses(
        responses, expected_status_and_error_messages, e2e_latencies
    )

    # 低优先级请求 (priority=0) 因被抢占，e2e 耗时应该小于高优先级请求 (priority=5)
    assert e2e_latencies[0] < e2e_latencies[1]
```

评论区精华

本次 PR 没有公开的 review 讨论，但 commit 历史显示了迭代过程：第一次提交只加了 asyncio.sleep(1.0)，第二次提交补上 ignore_eos=True 以进一步稳定时序，第三次提交进行代码结构优化。作者通过 10 次 CI 运行验证了修复的有效性。

- 暂无高价值评论线程

风险与影响

- 风险：变更范围仅限于单个测试函数的实现细节，未触及调度器或服务器核心逻辑。风险极低。唯一潜在风险是 asyncio.sleep(1.0) 的硬编码超时可能在极端慢的环境中仍然不足，但测试本身并不严格依赖精确时序，加上 ignore_eos 后时序容错性较高。
- 影响：直接消除了 test_priority_scheduling_preemption_below_threshold_validation 的 flaky 问题，提升 CI 可靠性。对系统其他部分无影响。
- 风险标记：暂无

关联脉络

- 暂无明显关联 PR