PR #1873 完整报告

THUDM/slime

[Fix] Use Ray ObjectRef await instead of asyncio.to_thread in distributed POST

合并时间：2026-04-28 10:35

原文链接：http://prhub.com.cn/THUDM/slime/pull/1873

执行摘要

- 一句话：移除分布式 POST 的线程池瓶颈，直接 await Ray ObjectRef
- 推荐动作：该 PR 虽改动极小，但值得所有使用分布式 POST 的团队关注。建议精读以理解 asyncio 与 Ray 集成的正确用法，并可作为后续异步化其他阻塞调用的参考。

功能与动机

分布式 POST 路径使用 asyncio.to_thread(ray.get, obj_ref)，其底层线程池隐含 min(32, cpu+4) 的并发上限，高并发下导致大量请求排队等待 OS 线程，产生人为尾延迟和响应突发。Ray 的 ObjectRef 本身就支持 asyncio await，可直接等待而不占用线程。

实现拆解

1. 定位瓶颈: 在 slime/utils/http_utils.py 的 post() 函数中，分布式分支通过 asyncio.to_thread(ray.get, obj_ref) 将 ray.get 提交到事件循环的默认线程池，受限于 ThreadPoolExecutor 的固定线程数。
2. 直接替换: 将 return await asyncio.to_thread(ray.get, obj_ref) 改为 return await obj_ref，利用 Ray 的 asyncio 集成——ObjectRef 可作为 asyncio.Future 等待，由 Ray 内部通过 loop.call_soon_threadsafe 唤醒协程，不消耗 OS 线程。
3. 保持其他逻辑不变: 异常处理、回退本地路径、非分布式路径均未改动。

关键文件：

- slime/utils/http_utils.py（模块 网络工具；类别 source；类型 core-logic）：本次变更的唯一文件，修改了分布式 POST 路径的核心等待逻辑。

关键符号：post

关键源码片段

slime/utils/http_utils.py

本次变更的唯一文件，修改了分布式 POST 路径的核心等待逻辑。

```
# slime/utils/http_utils.py (post 函数分布式分支 )

async def post(url, payload, max_retries=60, headers=None):
    # If distributed mode is enabled and actors exist, dispatch via Ray.
```

```
    if _distributed_post_enabled and _post_actors:
        try:
            import ray

            actor = _next_actor()
            if actor is not None:
                # Await the Ray ObjectRef directly. The previous
                # `asyncio.to_thread(ray.get, obj_ref)` blocked an OS thread
                # from the default ThreadPoolExecutor (capped at
                # `min(32, cpu+4)`), which becomes a hard upper bound on the
                # number of in-flight POSTs that can be waited on in parallel
                # and produces large tail latencies under high concurrency.
                obj_ref = actor.do_post.remote(url, payload, max_retries, headers=headers)
                return await obj_ref # 不再占用线程池，直接等待 Ray 的异步回调
        except Exception as e:
            logger.info(f"[http_utils] Distributed POST failed, falling back to local: {e} (url={url})")
            # fall through to local

    return await _post(_http_client, url, payload, max_retries, headers=headers)
```

评论区精华

该 PR 没有任何 review 评论或讨论。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。变更仅替换了一行核心调用，且利用了 Ray 官方文档推荐的 asyncio API。主要风险在于：若使用者不需要 asyncio 环境（如纯同步上下文），则 await 可能不适用；但当前函数已被声明为 async，因此变更安全。此外，回退机制保留，异常时仍可降级到本地 POST。
- 影响：影响范围局限于启用 --use-distributed-post 的部署场景。预期收益：在高并发 POST 请求下，消除线程池瓶颈，降低尾延迟，提高吞吐量。非分布式路径及其他功能无影响。
- 风险标记：低风险，核心路径变更但改动极小

关联脉络

- 暂无明显关联 PR