PR #6052 完整报告

verl-project/verl

[fully_async] fix: avoid blocking ray.get inside async actor methods

合并时间：2026-04-20 13:14

原文链接：http://prhub.com.cn/verl-project/verl/pull/6052

执行摘要

- 一句话：修复完全异步训练器中同步 Ray 调用阻塞事件循环的问题。
- 推荐动作：该 PR 值得精读，特别是对于使用 Ray 异步 Actor 的开发者。关注点包括： 1) 如何正确地将同步 Ray 调用迁移到异步以避免阻塞； 2) 使用 asyncio.wrap_future 处理 Ray 远程 future 的模式； 3) 讨论中关于 None 处理的设计权衡，展示了实际开发中边界条件处理的优先级决策。

功能与动机

根据 PR body 描述，在异步 Ray Actor 的 async def 方法中使用同步 ray.get 会阻塞 Actor 的事件循环，导致 Ray 运行时在每个受影响位置打印警告“Using blocking ray.get inside async actor. This blocks the event loop...”。同时，message_queue.py 中已标记同步助手为“deprecated, use instead”，且异步变体已实现。因此需要修复这些阻塞调用以消除警告并遵循异步最佳实践。

实现拆解

1. 导入 asyncio 模块：在 fully_async_trainer.py 头部添加 import asyncio，为后续使用 asyncio.wrap_future 提供支持。
2. 替换消息队列同步调用为异步调用：在 fully_async_trainer.py 的 _get_samples_from_queue 方法中，将 self.message_queue_client.get_sample_sync() 替换为 await self.message_queue_client.get_sample()；在 fully_async_rollouter.py 的 _should_pause_generation 和 get_statistics 方法中，将 self.message_queue_client.get_statistics_sync() 替换为 await self.message_queue_client.get_statistics()。
3. 替换 Ray 远程调用的同步获取为异步获取：在 fully_async_trainer.py 的 _fit_update_weights 方法中，将 ray.get(self.rollouter.reset_staleness.remote()) 替换为 await asyncio.wrap_future(self.rollouter.reset_staleness.remote().future())；在 _fit_validate 方法中，将 ray.get(val_future) 替换为 await asyncio.wrap_future(val_future.future())。
4. 测试验证：PR body 提到在 1×8 H200 上使用 Qwen3-4B-Instruct-2507 进行了 50 步完全异步 PPO 冒烟测试，预补丁和后补丁均成功，且警告消失，性能指标在运行间噪声范围内。

关键文件：

- verl/experimental/fully_async_policy/fully_async_trainer.py（模块 异步策略；类别 source；类型 core-logic；符号 _get_samples_from_queue, _fit_update_weights, _fit_validate）：修复 Trainer 异步 Actor 中多个阻塞调用，包括消息队列采样和 Ray 远程调用，是消除警告的核心文件。
- verl/experimental/fully_async_policy/fully_async_rollouter.py（模块 异步策略；类别 source；类型 core-logic；符号 _should_pause_generation, get_statistics）：修复 Rollouter 异步 Actor 中消息队列统计的同步调用，确保监控循环不阻塞。

关键符号：_get_samples_from_queue, _fit_update_weights, _fit_validate, _should_pause_generation, get_statistics

关键源码片段

verl/experimental/fully_async_policy/fully_async_trainer.py

修复 Trainer 异步 Actor 中多个阻塞调用，包括消息队列采样和 Ray 远程调用，是消除警告的核心文件。

```
async def _get_samples_from_queue(self) -> tuple[None, None] | tuple[int, Any]:
    """
    从消息队列获取样本并组成gen_batch_output
    使用循环持续收集样本直到足够数量
    """
    print(
        f"[FullyAsyncTrainer] Requesting {self.required_samples} samples from queue",
        flush=True,
    )

    # 使用简单循环调用 get_sample 收集样本
    consumer_start = time.time()
    queue_samples = []
    queue_len = 0
    while len(queue_samples) < self.required_samples:
        # 获取单个样本并等待直到有样本或收到 None
        # 修复：将同步调用 get_sample_sync() 替换为异步调用 get_sample()，避免阻塞事件循环
        sample, queue_len = await self.message_queue_client.get_sample()

        if sample is None:
            print(
                f"[FullyAsyncTrainer] Detected termination signal (None), stopping sample collection.
                "
                f"Collected {len(queue_samples)}/{self.required_samples} samples"
            )
            break

        queue_samples.append(sample)

        if len(queue_samples) % 64 == 0:
```

```
            print(
                f"[FullyAsyncTrainer] Collected {len(queue_samples)}/{self.required_samples}
                samples. "
                f"mq_len: {queue_len}"
            )

        consumer_end = time.time()

        if not queue_samples or len(queue_samples) < self.required_samples:
            print("[FullyAsyncTrainer] not enough samples collected after loop")
            return None, None
```

评论区精华

reviewer gemini-code-assist[bot] 指出在 fully_async_trainer.py 第 251 行，get_sample() 方法可能在消息队列关闭且为空时返回 None（标量），尝试解包 None 到 sample, queue_len 会引发 TypeError。建议在解包前安全处理潜在的 None 返回值以避免崩溃。作者 yxs 回复“out of scope”，认为此问题超出本 PR 范围，未采纳建议。

- get_sample() 返回 None 时的解包错误处理 (correctness): 作者未采纳建议，认为该边缘情况处理不属于本次修复范围。

风险与影响

- 风险：
  1. 回归风险：将同步调用改为异步可能引入竞态条件或死锁，但变更仅限于调用方式，不改变核心逻辑，风险较低。
  2. 正确性风险：get_sample() 返回 None 时的解包错误未被修复，在消息队列关闭场景下可能引发 TypeError 崩溃，但根据作者回复，此场景被视为边缘情况且可能由其他机制处理。
  3. 兼容性风险：依赖 asyncio.wrap_future 和 Ray 异步 API，要求 Ray 版本支持这些特性，但鉴于项目已广泛使用异步 Actor，风险可控。
- 影响：
  1. 对系统的影响：消除了阻塞事件循环的警告，提升了异步 Actor 的响应性和吞吐量，有助于完全异步训练流程的稳定性。
  2. 对用户的影响：用户将不再看到相关警告，但行为无变化，不影响训练结果。
  3. 对团队的影响：强化了异步编程规范，为后续完全异步功能开发提供了更清晰的范例。
     - 风险标记：异步编程风险，边缘情况未处理

关联脉络

- PR #6046 [fully_async] fix: preserve per-iteration routed_experts on partial rollout resume: 同属 fully_async 模块，涉及完全异步训练中的 rollout 恢复逻辑，可能共享类似异步编程模式。

- PR #6029 [fully_async] fix: replace routed_experts on partial rollout resume i…: 同属 fully_async 模块，修复完全异步策略中的 rollout 问题，上下文相关。
- PR #6041 [rollout] fix: RM sleep/wake teacher replicas: 涉及 rollout 和异步逻辑调整，可能影响相关组件。