PR #6166 完整报告

verl-project/verl

[fully_async] fix: fix custom reward function register on async mode

合并时间：2026-04-27 15:19

原文链接：http://prhub.com.cn/verl-project/verl/pull/6166

执行摘要

- 一句话：修复全异步模式下自定义奖励函数未注册问题
- 推荐动作：建议精读，并确认 migrate_legacy_reward_impl 的调用时机是否满足所有场景。若后续对配置检测逻辑有改动，需同步调整该调用位置。

功能与动机

PR body 明确指出“fix custom_reward_function not register in fully-async mode”。在全异步模式下，自定义奖励函数依赖于新版 reward 配置格式，但入口脚本未能将旧版 reward_model 配置（如 reward_model.reward_func）自动迁移到新格式，导致注册失败。

实现拆解

1. 导入迁移函数：在 verl/experimental/fully_async_policy/fully_async_main.py 中增加 from verl.experimental.reward_loop import migrate_legacy_reward_impl 导入。
2. 调用迁移：在 main() 函数末尾、run_ppo() 调用之前，插入 config = migrate_legacy_reward_impl(config)，将旧版 reward_model 键下的配置合并到新版 reward 键下。 该函数在其他入口（如非异步 main_ppo）中已被使用，此处仅补齐对全异步入口的支持。

关键文件：

- verl/experimental/fully_async_policy/fully_async_main.py（模块 全异步入口；类别 source；类型 entrypoint；符号 migrate_legacy_reward_impl）：唯一修改的文件，添加了导入 migrate_legacy_reward_impl 的语句并调用该函数，使全异步入口支持自定义奖励函数注册。

关键符号：migrate_legacy_reward_impl

关键源码片段

verl/experimental/fully_async_policy/fully_async_main.py

唯一修改的文件，添加了导入 migrate_legacy_reward_impl 的语句并调用该函数，使全异步入口支持自定义奖励函数注册。

```
# verl/experimental/fully_async_policy/fully_async_main.py
import os
```

```
import socket
import threading
from pprint import pprint

import hydra
import ray
from omegaconf import OmegaConf

from verl.experimental.fully_async_policy.fully_async_rollouter import FullyAsyncRollouter
from verl.experimental.fully_async_policy.fully_async_trainer import FullyAsyncTrainer
from verl.experimental.fully_async_policy.message_queue import MessageQueue,
MessageQueueClient
# 导入奖励配置迁移函数，确保自定义奖励函数能在新版配置格式中注册
from verl.experimental.reward_loop import migrate_legacy_reward_impl
from verl.experimental.separation.utils import create_resource_pool_manager, create_role_
worker_mapping
from verl.trainer.ppo.utils import Role
from verl.utils.device import auto_set_device
from verl.utils.fs import copy_to_local

@ray.remote(num_cpus=1)
class FullyAsyncTaskRunner:
    # ... （类定义不变）

@hydra.main(config_path="config", config_name="fully_async_ppo_trainer", version_base=None)
def main(config):
    from verl.trainer.main_ppo import run_ppo

    if not hasattr(config, "async_training"):
        raise RuntimeError("must set async_training config")

    assert config.async_training.use_trainer_do_validate is False

    from verl.trainer.ppo.utils import need_reward_model
    if need_reward_model(config) and config.async_training.use_trainer_do_validate:
        raise NotImplementedError(...)

    from time import time
    start_time = time()
    auto_set_device(config)
    config.actor_rollout_ref.rollout.nnodes = config.rollout.nnodes
    config.actor_rollout_ref.rollout.n_gpus_per_node = config.rollout.n_gpus_per_node
    # 调用迁移函数：将旧版 reward_model 配置合并到新版 reward 键下
    config = migrate_legacy_reward_impl(config)
    run_ppo(config, task_runner_class=FullyAsyncTaskRunner)
    print(f"total time: {time() - start_time:.2f} seconds")

if __name__ == "__main__":
    main()
```

评论区精华

gemini-code-assist[bot] 提出迁移调用应提前到 main() 开头（在 need_reward_model(config) 之前），但 wuxibin89 最终 approve 了当前写法。当前实现确保迁移发生在所有配置依赖的逻辑之后、run_ppo 之前，可能经过作者确认不影响 need_reward_model 等后续逻辑的正确性。

- migrate_legacy_reward_impl 调用时机 (design): wuxibin89 批准了当前写法，未采纳位置调整建议。当前调用位于 run_ppo 之前，足以保证奖励模型注册正确。

风险与影响

- 风险：时机风险：若 need_reward_model 或其他早期函数依赖于新版 reward 配置，则当前晚迁移可能导致检测错误。但 PR 中并未修改那些函数，且现有逻辑仍按旧版键获取奖励配置，因此风险较低。回归风险：几乎为零，改动仅影响全异步入口，不改变函数逻辑。
- 影响：影响范围：仅限于全异步训练模式（fully_async_main.py），使该模式下自定义奖励函数能正确注册。影响程度：对使用全异步框架并依赖自定义奖励函数的用户至关重要；对同步训练或其他模式无影响。
- 风险标记：配置迁移时机潜在依赖顺序

关联脉络

- PR #6044 [fully_async, reward] feat: enable GenRM/DisRM support in fully async training: PR #6044 引入了全异步模式下独立奖励模型的支持，与本 PR 的自定义奖励函数注册属于同一功能线。