PR #6461 完整报告

verl-project/verl

[fully_async] feat: fully async profiling

合并时间：2026-05-26 15:58

原文链接：http://prhub.com.cn/verl-project/verl/pull/6461

执行摘要

- 一句话：为 fully_async 模式添加 profiling 支持，统一步数定义
- 推荐动作：建议精读。该 PR 是 fully_async 功能完善的重要一步，其中关于 step 概念统一的设计值得关注（区分 global_steps 与 current_param_version）。代码改动集中在少数文件，逻辑清晰，适合作为异步模式下 profiling 集成的参考实现。

功能与动机

PR body 指出："The concept of step is ambiguous in fully async mode. In rollout, rollout_step refers to the generation of each prompt. In the trainer, train_step means executing one mini-batch step. … Therefore, we follow the step definition of sync mode, and regard the completion of one update_weights as the end of one training step." 为了与同步模式的 profiling 体验对齐，需要将 profiling 绑定到参数更新次数而非 mini-batch step。

实现拆解

1. 扩展父类 profiling 方法：在 verl/experimental/separation/ray_trainer.py 中，为 _fit_start_profile 和 _fit_stop_profile 增加可选的 should_profiler 参数。若外部传入该参数，则忽略内部 prev_step_profile / curr_step_profile 逻辑，直接使用传入值决定是否启停 profiler。这为子类提供了灵活的 profiling 控制点。
2. 调整 FullyAsyncTrainer 初始化与 step 判断：在 verl/experimental/fully_async_policy/fully_async_trainer.py 的 fit 方法中，将 curr_step_profile 初始化为 False（不再依赖 global_steps 计算）。在 fit_step 中，使用 steps = self.config.global_profiler.steps 和 (self.current_param_version + 1) in steps 计算 should_profile，并传递给 _fit_start_profile 和 _fit_stop_profile。
3. 在权重更新前后控制 rollout profiling：在 _fit_update_weights 方法中，在调用 update_weights 之前，根据当前 current_param_version 是否在 profile steps 中，调用 self.rollouter._stop_profiling.remote() 停止旧 step 的 profiling；在 update_weights 之后，根据下一个 current_param_version + 1 是否在 profile steps 中，调用 self.rollouter._start_profiling.remote() 启动新 step 的 profiling。
4. 在 Rollouter 中实现 profiling 启停接口：在 verl/experimental/fully_async_policy/fully_async_rollouter.py 中新增 _start_profiling 和 _stop_profiling 两个 async 方法，分别委托给 self.llm_server_manager.start_profile() 和 self.llm_server_manager.stop_profile()，

复用 agent loop 已有的 profiling 能力。

5. 更新 Ascend profiling 文档：在 docs/ascend_tutorial/dev_guide/performance/ascend_profiling_en.rst 和 ascend_profiling_zh.rst 中添加「Fully Async Policy 模式说明」，强调 global_profiler.steps 指每次 update_weights 后的 step，与同步模式一致，并指出该模式复用 AgentLoop 的采集能力，注意事项相同。

关键文件：

- verl/experimental/separation/ray_trainer.py（模块 分离训练器；类别 source；类型 core-logic；符号 _fit_start_profile, _fit_stop_profile）：修改父类 profiling 方法 _fit_start_profile 和 _fit_stop_profile，添加 should_profiler 参数以支持外部控制，这是实现子类定制的基础。
- verl/experimental/fully_async_policy/fully_async_trainer.py（模块 异步训练器；类别 source；类型 core-logic；符号 fit, fit_step, _fit_update_weights）：核心变更文件，调整 fit 和 fit_step 中的 profile 初始化与触发逻辑，并在 _fit_update_weights 中集成 rollouter profiling 启停。
- verl/experimental/fully_async_policy/fully_async_rollouter.py（模块 异步回滚器；类别 source；类型 core-logic；符号 _start_profiling, _stop_profiling）：新增 rollouter 端 profiling 接口，复用 llm_server_manager 的 start/stop_profile 方法。
- docs/ascend_tutorial/dev_guide/performance/ascend_profiling_en.rst（模块 Ascend 文档；类别 docs；类型 documentation）：更新英文文档，说明 fully_async 模式下 global_profiler.steps 的语义。
- docs/ascend_tutorial/dev_guide/performance/ascend_profiling_zh.rst（模块 Ascend 文档；类别 docs；类型 documentation）：更新中文文档，与英文文档同步说明。

关键符号：_fit_start_profile, _fit_stop_profile, _start_profiling, _stop_profiling, fit, fit_step, _fit_update_weights

关键源码片段

verl/experimental/separation/ray_trainer.py

修改父类 profiling 方法 _fit_start_profile 和 _fit_stop_profile，添加 should_profiler 参数以支持外部控制，这是实现子类定制的基础。

```
# verl/experimental/separation/ray_trainer.py

# 修改前：_fit_start_profile 无参数，自行决定 profile 开关
# 修改后：接受可选的 should_profiler，若不为 None 则直接使用，
# 否则使用原有的基于 prev/curr_step_profile 的逻辑。
def _fit_start_profile(self, should_profiler=None):
    timing_raw = self.timing_raw
    if should_profiler is not None:
        # 外部传入：直接覆盖自我状态
        self.curr_step_profile = should_profiler
    with marked_timer("start_profile", timing_raw):
        if should_profiler is None:
```

```
            # 原有判断：连续 profile 模式下判断起始条件
            do_profile = (
                not self.prev_step_profile and self.curr_step_profile
                if self.config.global_profiler.profile_continuous_steps
                else self.curr_step_profile
            )
        else:
            do_profile = should_profiler
        self._start_profiling(do_profile)

    def _fit_stop_profile(self, should_profiler=None):
        timing_raw = self.timing_raw
        with marked_timer("stop_profile", timing_raw):
            if should_profiler is None:
                # 原有判断：计算下一个 step 是否需要 profile
                self.next_step_profile = (
                    self.global_steps + 1 in self.config.global_profiler.steps
                    if self.config.global_profiler.steps is not None
                    else False
                )
                do_profile = (
                    self.curr_step_profile and not self.next_step_profile
                    if self.config.global_profiler.profile_continuous_steps
                    else self.curr_step_profile
                )
            else:
                do_profile = should_profiler
            self._stop_profiling(do_profile)
            self.prev_step_profile = self.curr_step_profile
            if should_profiler is None:
                self.curr_step_profile = self.next_step_profile
```

verl/experimental/fully_async_policy/fully_async_trainer.py

核心变更文件，调整 fit 和 fit_step 中的 profile 初始化与触发逻辑，并在 _fit_update_weights 中集成 rollouter profiling 启停。

```
    # verl/experimental/fully_async_policy/fully_async_trainer.py

    async def fit(self):
        ...
        self.global_steps += 1
        self.prev_step_profile = False
        # 子类中不再依赖 global_steps 计算 curr_step_profile
        self.curr_step_profile = False # 修改前：基于 global_steps 初始化
        self.next_step_profile = False
        ...

    async def fit_step(self, batch_dict: dict = None):
        ...
```

```
    # 使用 (current_param_version + 1) 对应"即将完成的这次 update_weights"
    steps = self.config.global_profiler.steps
    should_profile = steps is not None and (self.current_param_version + 1) in steps
    self._fit_start_profile(should_profiler=should_profile) # 传递外部定义
    with marked_timer("step", self.timing_raw):
        ...
        await self._fit_update_weights()
        ...
    self._fit_stop_profile(should_profiler=should_profile)
    ...

async def _fit_update_weights(self):
    if self.local_trigger_step != 1:
        return
    # 在参数同步之前停止当前 step 的 rollout profiling
    steps = self.config.global_profiler.steps
    last_profiler_step = self.current_param_version
    if steps is not None and last_profiler_step in steps:
        await asyncio.wrap_future(self.rollouter._stop_profiling.remote().future())

    with marked_timer("timing_s/param_sync", self.timing_raw):
        await self.checkpoint_manager.update_weights(global_steps=self.current_param_version)
        ...

    # 在参数同步之后启动下一个 step 的 rollout profiling
    profiler_step = last_profiler_step + 1
    if steps is not None and profiler_step in steps:
        await asyncio.wrap_future(self.rollouter._start_profiling.remote().future())
    ...
```

评论区精华

1. gemini-code-assist[bot] 对 agent_loop.py 的评论：指出 generate 方法中使用了未定义的 kwargs 变量和缺少 DiffusionOutput 导入。但该文件最终未包含在 PR 变更集中，因此这两个问题未在本次 PR 中实际引入或解决。
2. gemini-code-assist[bot] 对 fully_async_rollouter.py 的评论：指出 get_curr_step_profile 方法中访问 self.config.global_profiler.rollout_steps 可能因配置键不存在而引发 AttributeError。但在最终提交的代码中，get_curr_step_profile 方法并未出现，取而代之的是在 _fit_update_weights 中直接使用 global_profiler.steps。因此该问题已被隐式解决。

最终审批者 wuxibin89 直接 approved，未进一步评论。

- agent_loop.py 中 generate 方法的潜在缺陷 (correctness): 该文件未包含在最终 PR 变更集中，因此问题未实际出现或已在其他地方修复。
- fully_async_rollouter.py 中配置键 rollout_steps 的合法性 (correctness): 最终代码中删除了 get_curr_step_profile 方法，改为直接使用 global_profiler.steps，该问题不再存在。

风险与影响

- 风险：
  1. 异步时序风险：在 _fit_update_weights 中，profiling 的 start/stop 与实际的参数同步之间可能存在竞态条件。如果 rollouter 的 _stop_profiling 和 _start_profiling 调用出现在网络或队列延迟之后，可能导致 profiling 数据与预期 step 不匹配。目前未看到显式的同步等待或回调保障。
  2. 缺少测试覆盖：PR 未包含任何新的测试用例，仅依赖手动的端到端验证（如 PR body 所示）。配置文件路径或边界情况（如 profile_continuous_steps 启用时）无法在 CI 中持续验证。
  3. 依赖内部接口：llm_server_manager.start_profile() 和 stop_profile() 是 agent loop 的内部方法，若未来 agent loop 重构，该接口可能失效或改变语义。
  4. 文档误导风险：中文文档中 global_profiler.steps 代表每一轮 update_weights 后的 step 的表述可能不够精确，用户容易混淆 mini-batch step 与 weight-update step 的计数起点。 - 影响：用户影响：开发者和用户在 fully_async 模式训练时，现在可以通过与同步模式相同的配置方式启用 global_profiler.steps 进行性能剖析，降低了使用门槛。系统影响：新增的 profiling 调用会增加训练流程中的异步 RPC 开销，但仅在指定 step 触发，整体开销可控。团队影响：该 PR 清晰定义了异步模式下的 step 语义，为后续 metrics、checkpoint 等与 step 相关的功能提供了统一时间基准。
- 风险标记：异步时序敏感，缺少测试覆盖，依赖 llm_server_manager 接口

关联脉络

- PR #6457 [fully_async,doc] feat: rm future plans, almost all completed.: 同为 fully_async 模块的文档更新，但侧重点不同。本 PR 新增 profiling 文档，6457 移除未来计划。
- PR #6423 [rollout] feat: add Trackio rollout trace logging: 同样增强 rollout 可观测性，但使用不同的 trace 后端。本 PR 补充了 profiling 集成，两者共同提升异步模式的可观测性。