

PR #1843 完整报告

radixark/miles

Wire up InferenceController and RolloutExecutor and make the controller API async

合并时间: 2026-09-04 12:20

原文链接: <http://prhub.com.cn/radixark/miles/pull/1843>

执行摘要

- 一句话: 将 RolloutManager 拆分为 InferenceController 与 RolloutExecutor, 并将控制器 API 异步化。
- 推荐动作: 这是一个需要精读的架构级重构 PR。建议关注: 1) InferenceController 异步 API 的设计及其对训练循环时序的影响; 2) RolloutComponents 的构建顺序如何确保依赖 (如路由器地址) 正确传递; 3) 测试如何验证新组件间的职责边界。鉴于其高风险和跨模块影响, 合并后需密切监控所有训练作业。

功能与动机

本 PR 是 Issue #1837 追踪的一系列重构增强的一部分, 其直接目标是完成对原 **RolloutManager** 的职责拆分 (见关联 PR #1842)。动机在于:

- 1) 解耦推理引擎的生命周期管理 (InferenceController) 与数据采样执行 (RolloutExecutor), 使职责更清晰;
- 2) 将控制器 (管理引擎) 的 API 改为异步, 以消除同步阻塞 (如 **ray.get**), 为后续的异步工作流和多训练器场景做准备。

实现拆解

1. 核心组件创建重构 (miles/ray/placement_group.py): 将原有的 create_rollout_manager 函数重构为 create_rollout_components, 并引入 RolloutComponents 命名元组。该函数现在同步创建一个驱动器端的 InferenceController 实例, 并异步创建一个作为 Ray actor 的 RolloutExecutor。随后, 它将评估舰队 (eval_fleet) 从控制器传递给执行器, 并将原先针对旧 RolloutManager 的同步 ray.get 调用 (如 check_weights, offload_kv) 替换为针对新异步控制器的直接 await 调用。
2. 训练组分配接口更新 (miles/ray/placement_group.py, miles/ray/train/group.py, miles/ray/actor_group.py): 修改 allocate_train_group 函数及其调用方, 使其接收 inference_controller 和 rollout_executor 两个参数, 替代原先单一的 rollout_manager。同时, 将训练组内广播的 set_rollout_manager 方法统一重命名为 set_rollout_executor, 并在训练 actor 的初始化逻辑中适配新参数。
3. 训练入口与循环适配 (train.py, train_async.py): 更新主训练脚本 train.py 及其异步版本 train_async.py, 将创建和使用 rollout_manager 的逻辑替换为使用新的 inference_controller 和 rollout_executor。关键的训练循环步骤 (如准备评估、准备采样、执行生成 / 评估、卸载权重) 现在分别调用控制器或执行器的对应方法, 体现了职责分离。

4. 控制器 API 异步化与职责微调 (miles/ray/rollout/inference_controller.py): 将 InferenceController 的多个方法 (如 prepare_eval, dispose, clear_updatable_has_new_engines, health_monitoring_pause) 从同步改为 async, 使其能够 await 内部的子操作。同时, 从控制器初始化流程中移除了 init_http_client 的调用, 因为该职责现在归属 RolloutExecutor。
5. 测试套件同步更新与新增 (tests/fast/ray/...): 大量测试文件被更新以反映新的 API。例如, test_rollout_manager.py 被重命名为 test_inference_controller.py 并调整测试焦点。新增了 test_rollout_executor.py 和 test_placement_group.py 以专门覆盖新组件。同时, 新增了 tests/fast/ray/fake_rollout_executor.py 模块以提供测试用的模拟对象。

关键文件:

- miles/ray/placement_group.py (模块 组件创建; 类别 source; 类型 core-logic; 符号 create_training_models, create_rollout_components, RolloutComponents, allocate_train_group): 核心组件创建逻辑重构的入口点, 定义了新的 RolloutComponents 结构和异步创建流程, 是理解整个变更的关键。
- miles/ray/rollout/inference_controller.py (模块 推理控制器; 类别 source; 类型 entrypoint; 符号 prepare_eval, dispose, clear_updatable_has_new_engines, health_monitoring_pause): 核心变更之一, 将控制器 API 异步化, 移除了 HTTP 客户端初始化等职责。
- miles/ray/train/group.py (模块 训练组; 类别 source; 类型 core-logic; 符号 set_rollout_executor, update_weights): 训练组逻辑适配新接口, 将 set_rollout_manager 重命名为 set_rollout_executor, 并更新权重更新流程。

关键符号: create_rollout_components, create_training_models, set_rollout_executor, InferenceController.init, InferenceController.health_monitoring_pause, train

关键源码片段

miles/ray/placement_group.py

核心组件创建逻辑重构的入口点, 定义了新的 RolloutComponents 结构和异步创建流程, 是理解整个变更的关键。

```
# from miles/ray/placement_group.py
from typing import NamedTuple
from miles.ray.rollout.inference_controller import InferenceController
from miles.ray.rollout.rollout_executor import RolloutExecutor

class RolloutComponents(NamedTuple):
    """新建的命名元组, 封装拆分后的两个核心组件。"""
    inference_controller: InferenceController # 在驱动器进程中运行, 管理引擎生命周期。
    rollout_executor: ray.actor.ActorHandle # 作为 Ray actor 运行, 执行数据采样。
    num_rollout_per_epoch: int | None

async def create_rollout_components(args, pg) -> RolloutComponents:
    """创建推理控制器和执行器, 替代原有的 create_rollout_manager。"""
    # 1. 在驱动器进程中同步创建控制器。
```

```

inference_controller = InferenceController(args, pg)

# 2. 创建执行器作为远程 Ray actor。
rollout_executor = RolloutExecutor.options(
    num_cpus=1, num_gpus=0, **(compute_ray_pin_head_options() if args.pin_rollout_
    manager_to_head else {}
).remote(args) # 注意：此处参数已简化，不再传递 pg。

# ... ( 计算 num_rollout_per_epoch) ...

# 3. 将评估舰队从控制器传递给执行器。
await rollout_executor.set_eval_fleet.remote(inference_controller.eval_fleet)

# 4. 原先针对旧 manager 的同步调用，改为针对新 controller 的异步调用。
if args.check_weight_update_equal:
    await inference_controller.check_weights(action="snapshot")
    await inference_controller.check_weights(action="reset_tensors", skip_list=args.check_
    weight_update_skip_list)

if args.offload_rollout:
    if args.colocate_memory_peak_device == "gpu":
        await inference_controller.offload_kv()
    else:
        await inference_controller.offload()

return RolloutComponents(
    inference_controller=inference_controller,
    rollout_executor=rollout_executor,
    num_rollout_per_epoch=num_rollout_per_epoch,
)

```

miles/ray/rollout/inference_controller.py

核心变更之一，将控制器 API 异步化，移除了 HTTP 客户端初始化等职责。

```

# from miles/ray/rollout/inference_controller.py
# 关键方法签名变更，体现异步化。

async def prepare_eval(self):
    """异步化：为评估周期做准备。"""
    self._health_monitoring_resume()

async def dispose(self):
    """异步化：清理资源。"""
    for monitor in self._health_monitors:
        monitor.stop()

async def clear_updatable_has_new_engines(self):
    """异步化：清除引擎的新状态标记。"""
    srv = self._get_updatable_server()

```

```
if srv:
    srv.clear_has_new_engines()

# offload 方法也变为 await self.health_monitoring_pause()
# health_monitoring_pause 自身也改为 async。
```

miles/ray/train/group.py

训练组逻辑适配新接口，将 `set_rollout_manager` 重命名为 `set_rollout_executor`，并更新权重更新流程。

```
# from miles/ray/train/group.py
# 在 update_weights 中，原先的 ray.get(rollout_manager.X.remote()) 变为 await inference_
controller.X()
async def update_weights(self, rollout_id: int | None = None):
    # ... 获取引擎和锁（现在是控制器的异步方法）。
    info = await self._inference_controller.get_updatable_engines_and_lock()
    await self._inference_controller.health_monitoring_pause()
    # ... 执行权重更新 ...
    await self._inference_controller.clear_updatable_has_new_engines()
    # ...

# 将广播方法名统一改为 set_rollout_executor
async def set_rollout_executor(self):
    await asyncio.gather(*[cell.set_rollout_executor() for cell in self._cells])
```

评论区精华

本 PR 自身的 Review 评论区为空，但其关联的 Issue #1837 的评论中包含了重要的后续修复讨论：

- 进程边界正确性 (Process-Boundary Correctness): 评论者 guapisolo 指出，本 PR 在拆分组件时引入了一个状态不一致的 bug（控制器侧和执行器侧的 `EvalFleet` 状态需分离）。建议将一个修复此进程边界问题的提交（源自 PR #2614）作为本 PR 的直接后续，以确保拆分后的组件间通过数据（而非状态拷贝）进行通信。
- 审计日志兼容性 (Replay Compatibility): 同一条评论还提到，需要保留对旧审计日志 JSONL 格式（`component="rollout_manager"`）的兼容性，或提供明确的迁移路径，因为后续 PR #2503 目前只接受 `rollout_executor` 组件名。
 - 进程边界正确性（`EvalFleet` 状态分离）（`correctness`）: 需要后续修复，确保组件间通过数据而非状态拷贝进行通信。
 - 审计日志（JSONL）向后兼容性（`other`）: 需要迁移路径或别名，避免旧数据丢失。

风险与影响

- 风险：高风险：这是一个影响训练核心路径（权重同步、数据生成、引擎生命周期管理）的大规模重构。虽然有配套测试，但异步 API 的引入和组件交互方式的改变可能引入难以预见的时序问题或竞态条件，尤其是在故障恢复（FT）路径和权重更新期间。中风险：`InferenceController`（驱动器端）与 `RolloutExecutor`（actor 端）的新进程边界可能引入

状态同步问题，如 Issue 评论中提及的 `EvalFleet` 状态不一致问题。低风险：测试代码的适配和重命名，以及配置参数的传递变更。

- 影响：对用户：本变更不直接影响最终用户（如使用开源 Miles 库的用户），因为它属于内部架构重构。然而，它为未来的功能（如完全异步训练、多训练器支持）铺平了道路。对系统：显著改变了 Miles 训练系统的内部架构。训练循环的执行流程、组件间的通信方式（从 Ray actor 的远程调用变为驱动器内直接异步调用）都发生了根本变化。对团队：开发团队需要理解新的双组件架构和异步 API。后续的 PR（如 #1844 重命名 `generate` 方法）将基于此变更继续演进。
- 风险标记：核心路径变更，进程边界正确性风险，异步时序变化

关联脉络

- PR #1842 Split RolloutManager into InferenceController and RolloutExecutor: 本 PR 是此拆分 PR 的后续，完成了组件的接线（wiring）和控制器 API 的异步化。
- PR #1844 Rename RolloutExecutor.generate to get for future async and multi trainer scenarios: 紧随本 PR 之后，对 RolloutExecutor 的 API 进行进一步重命名，以更准确地反映其职责，同属 Issue #1837 重构系列。