PR #1716 完整报告

radixark/miles

Move fully-async rollout from examples into miles/rollout

合并时间：2026-08-01 05:28

原文链接：http://prhub.com.cn/radixark/miles/pull/1716

执行摘要

- 一句话：fully-async rollout 迁入核心库并清理 PYTHONPATH hack
- 推荐动作：值得快速阅读，重点看迁移模式和 PYTHONPATH 清理的边界处理。真正值得精读的是后续的类化 API 重写 PR（stacked follow-up）和 --fully-async 参数落地 PR，本 PR 可作为理解 fully-async 架构的入口文档。注意 gemini 提出的 ModuleNotFoundError 风险点，若仓库未强制安装 miles 包，建议合入后补一个环境验证。

功能与动机

PR body 明确指出 fully-async rollout 已被生产运行使用（swe-agent-v2 GLM-4.7 async 脚本），但实现一直放在 examples/fully_async/，每个启动脚本都要靠 PYTHONPATH hack 才能导入；而它消费的核心参数（如 --max-weight-staleness）早已定义在核心 arguments.py 中，消费方却在 examples 里，造成核心参数与实现割裂。此次移动是将其纳入核心库的第一步。

实现拆解

1. 文件移动：用 git mv 将 examples/fully_async/fully_async_rollout.py 移动到 miles/rollout/fully_async_rollout.py，保持 100% 内容不变（rename similarity 100%），保留 git blame 历史。
2. 引用路径更新：所有 --rollout-function-path fully_async_rollout.generate_rollout_fully_async 改为 miles.rollout.fully_async_rollout.generate_rollout_fully_async，涉及 examples/fully_async/run_qwen3_30b_a3b_fully_async.py、examples/fully_async/run-qwen3-4b-fully_async.sh、examples/swe-agent/run-glm47-flash-agentic-async.py。
3. 清理 PYTHONPATH hack：删除 run_qwen3_30b_a3b_fully_async.py 中动态拼接 fully_async_dir 的逻辑，PYTHONPATH 退化为纯 args.megatron_path；删除 run-glm47-flash-agentic-async.py 的 FULLY_ASYNC_DIR 常量并把它从 PYTHONPATH 中移除；run-qwen3-4b-fully_async.sh 的 RuntimeEnv 同理。
4. 文档同步：docs/examples/fully-async.md、docs/user-guide/fully-async.md、docs/developer/architecture.md、examples/fully_async/README.md 更新文件树与命令示例；train_async.py 和 examples/infra_features/random_async/random_async_rollout.py 的 docstring 引用同步更新。

5. 测试配套：本次为纯移动，未新增测试；PR body 说明 pre-commit run --all-files 通过，模块内容与导入路径保持兼容。

关键文件：

- examples/swe-agent/run-glm47-flash-agentic-async.py（模块 示例脚本；类别 source；类型 core-logic；符号 execute）：生产环境 swe-agent-v2 GLM-4.7 async 脚本，move 后引用路径切换为核心模块并删除 FULLY_ASYNC_DIR hack，是本次变更最重要的消费者。
- miles/rollout/fully_async_rollout.py（模块 核心 rollout；类别 source；类型 rename-or-move；符号 generate_rollout_fully_async, AsyncRolloutWorker, get_global_worker, stop_global_worker）：核心库新增的 fully-async rollout 模块，本次迁移的目标文件，后续类化重写的基础。
- examples/fully_async/run_qwen3_30b_a3b_fully_async.py（模块 示例脚本；类别 source；类型 dependency-wiring；符号 execute）：MoE 变体启动脚本，清理了动态 PYTHONPATH 拼接，保留核心路径切换。
- examples/fully_async/run-qwen3-4b-fully_async.sh（模块 示例脚本；类别 other；类型 configuration）：Qwen3-4B shell 启动脚本，同步更新 rollout-function-path 与 RuntimeEnv 的 PYTHONPATH。
- docs/examples/fully-async.md（模块 文档；类别 docs；类型 documentation）：示例文档，更新文件树与 rollout-function-path 示例，是用户感知迁移的主要渠道。

关键符号：generate_rollout_fully_async, AsyncRolloutWorker.start, get_global_worker, stop_global_worker, group_oldest_weight_version, _CachedWeightVersion.get, execute

关键源码片段

examples/swe-agent/run-glm47-flash-agentic-async.py

生产环境 swe-agent-v2 GLM-4.7 async 脚本，move 后引用路径切换为核心模块并删除 FULLY_ASYNC_DIR hack，是本次变更最重要的消费者。

```
# examples/swe-agent/run-glm47-flash-agentic-async.py 的 execute() 关键片段

def execute(args: ScriptArgs):
    # 完全异步模式下训练与 rollout 分属不同节点，权重更新不阻塞生成
    rollout_args = (
        # 核心路径：miles.rollout.fully_async_rollout 取代 examples 下的旧路径
        "--rollout-function-path miles.rollout.fully_async_rollout.generate_rollout_fully_async "
        f"--prompt-data {args.prompt_data} "
        "--input-key prompt "
        "--metadata-key metadata "
        "--rollout-shuffle "
        f"--num-rollout {args.num_rollout} "
        f"--rollout-batch-size {args.rollout_batch_size} "
        ...
    )

    miles_root = U.repo_base_dir
```

```
    extra_env_vars = {
        # 删除 FULLY_ASYNC_DIR 后的 PYTHONPATH，仓库根目录由 miles_root 提供
        "PYTHONPATH": f"{args.megatron_path}:{SCRIPT_DIR}:{miles_root}",
        "MILES_EXPERIMENTAL_ROLLOUT_REFACTOR": "1",
        ...
    }
```

miles/rollout/fully_async_rollout.py

核心库新增的 fully-async rollout 模块，本次迁移的目标文件，后续类化重写的基础。

```
# miles/rollout/fully_async_rollout.py（迁移后保持原样的核心骨架）

import asyncio
import atexit
import queue
import threading
import time
from miles.rollout.data_source import DataSource
from miles.rollout.sglang_rollout import GenerateState, generate_and_rm_group
from miles.utils.async_utils import run
from miles.utils.types import Sample

logger = logging.getLogger(__name__)


def group_oldest_weight_version(group: list[Sample]) -> int | None:
    """返回一组轨迹中所有 turn 的最小权重版本号，用于 staleness 判断。"""
    versions = [s.oldest_weight_version for s in group if s.oldest_weight_version is not None]
    return min(versions) if versions else None


class _CachedWeightVersion:
    """带 TTL 的权重版本查询，避免频繁打 /model_info 接口。"""
    def __init__(self, ttl: float = 1.0):
        self._ttl = ttl
        self._value: int | None = None
        self._last_query: float = 0.0

    async def get(self, args) -> int | None:
        now = time.monotonic()
        if self._value is not None and (now - self._last_query) < self._ttl:
            return self._value
        url = f"http://{args.sglang_router_ip}:{args.sglang_router_port}/model_info"
        try:
            async with aiohttp.ClientSession() as session:
                async with session.get(url, timeout=aiohttp.ClientTimeout(total=2)) as resp:
                    if resp.status == 200:
                        data = await resp.json()
                        self._value = int(data["weight_version"])
```

```
                    self._last_query = now
            except Exception as e:
                logger.debug(f"Failed to query engine weight version: {e}")
            return self._value


_cached_version = _CachedWeightVersion()

# 全局 worker 管理器：训练进程内只允许一个常驻异步 worker
_global_worker = None
_worker_lock = threading.Lock()


def get_global_worker(args, data_buffer: DataSource):
    """获取或创建全局 async worker，保证多轮训练间共享同一后台循环。"""
    global _global_worker
    with _worker_lock:
        if _global_worker is None or not _global_worker.worker_thread.is_alive():
            print("Creating new global async worker...")
            _global_worker = AsyncRolloutWorker(args, data_buffer)
            _global_worker.start()
        return _global_worker


def stop_global_worker():
    ... # atexit 注册，进程退出时清理后台线程
```

评论区精华

1. gemini-code-assist 指出 run_qwen3_30b_a3b_fully_async.py 移除本地目录后，PYTHONPATH 仅剩 args.megatron_path，若仓库根目录不在 Python 搜索路径或未安装 miles 包，导入 miles.rollout.fully_async_rollout 会 ModuleNotFoundError，建议加上 U.repo_base_dir（中等优先级，已采纳方向，但最终提交未显式加回）。
2. guapisolo 建议直接新增 --fully-async 参数，让所有用户知道存在 fully async 模式，而不是靠 --rollout-function-path 隐式指定（已接受，在后续 PR 处理，本 PR 原样合入）。
3. guapisolo 提出后续要补一个 dapo-math 的 fully async CI（已确认后续提交）。

- PYTHONPATH 裁剪导致的 ModuleNotFoundError 风险 (correctness): 未在最终提交中采纳该建议，保留了简化后的 PYTHONPATH，实际依赖 miles 包安装或默认路径。
- 建议新增 --fully-async 参数 (design): 一致同意，计划在后续 PR 中实现，本 PR 按原样合入。
- 补充 fully async CI (testing): 确认后续提交，本 PR 不包含测试。

风险与影响

- 风险：

1. 导入回归风险：run_qwen3_30b_a3b_fully_async.py 的 PYTHONPATH 不再显式包含仓库根目录，若在未安装 miles 包的环境中执行该脚本，导入 miles.rollout.fully_async_rollout 可能失败；review 已指出，但最终提交未加 U.repo_base_dir，属于遗留隐患。
2. 外部脚本兼容风险：文档和示例路径更新后，任何仍引用旧 examples/fully_async/fully_async_rollout 路径的脚本（包括用户私有脚本）将失效，但这是有意的 breaking change。
3. 启动脚本行为风险：run-qwen3-4b-fully_async.sh 中 RUNTIME_ENV_JSON 的 PYTHONPATH 也被裁剪，与 gemini 指出的问题同源，需在真实 GPU 环境验证；PR body 声称 'Pure mechanical move'，但 PYTHONPATH 语义实际发生了变化（从显式目录变为依赖包安装 / 默认路径）。 - 影响：对用户：fully-async rollout 的调用路径统一为 miles.rollout.fully_async_rollout.generate_rollout_fully_async，文档和示例同步更新，旧路径失效；生产 swe-agent-v2 脚本（run-glm47-flash-agentic-async.py）从 PYTHONPATH hack 中解脱，可依赖核心包安装。对系统：核心库 miles/rollout/ 新增一个生产级 rollout 模块，后续类化重构将在此基础上进行；--max-weight-staleness 等核心参数现在与消费方同处核心层，参数设计闭环。对团队：里程碑意义大于代码量——fully-async 从实验区正式进入核心代码库，后续 PR（--fully-async 参数、dapo-math CI、类化 API 重写）已排期。 - 风险标记：PYTHONPATH 裁剪风险，导入路径 breaking change，缺少测试覆盖

关联脉络

- PR #1916 (1/2) refactor(rollout): drop --generate-multi-samples and its per-turn sample semantics: 同为 rollout 重构系列，涉及多轮 rollout API 的调整，fully-async 模块后续要基于类化 rollout API 重写，属于同一演进线。
- PR #1759 (2/2) refactor(session): assemble training samples on the session server; records never leave it: session 采样链路重构，fully-async 的 rollout 数据流与之相关，同属 rollout/session 架构演进。
- PR #1829 fix: require explicit off-policy correction for async PPO training: train_async.py 的异步训练参数校验变更，与 fully-async 运行模式直接相关，后续 --fully-async 参数也可能涉及参数校验。
- PR #2013 [tito] Add the Inkling TITO family (Inkling / Inkling-Small): 同仓库后续会话 /rollout 演进，说明 rollout 模块仍在持续扩展，fully-async 迁入核心后便于统一维护。