PR #1717 完整报告

radixark/miles

Rewrite fully-async rollout as FullyAsyncRolloutFn on the class-based rollout API

合并时间：2026-08-01 12:26

原文链接：http://prhub.com.cn/radixark/miles/pull/1717

执行摘要

- 一句话：重写 fully-async rollout 为类式 API 并新增 --fully-async 开关
- 推荐动作：值得精读。这是 rollout 子系统从 example 堆栈升级为类 API 一等公民的关键一跃，设计决策密度高：错误显式化哲学（死 worker 优先于积压队列）、参数解析单一决策点 resolve_rollout_function_paths、"fully async 不服务 eval" 的构造性保证、权重版本缓存的失败节流、以及用 FakeDataSource 精确刻画状态机的测试 harness。建议重点读 miles/rollout/fully_async_rollout.py 的生产者 - 消费者结构与 miles/utils/arguments.py 的解析顺序；同时跟踪 _CachedWeightVersion 未修复的 review 意见和共享实例的并发隐患是否在后续 PR 处理。

功能与动机

PR body 明确说明：#1716 把 fully_async_rollout.py 从 examples/ 移入 miles/rollout/ 时是原样搬入，"What landed was still example-quality code on the legacy stack"——模块级全局 worker（_global_worker + threading.Lock + atexit）、私有线程事件循环、sglang_rollout 的 GenerateState 单例、到处 print、宽泛 try/except 静默丢数据。单例 semaphore 绑定到 worker 私有循环，导致 fully-async + eval 在结构上不可能（跨循环 RuntimeError）。此外 drain 重写了收集循环，标准路径的 --dynamic-sampling-filter-path（swe-agent 示例实际在用）等被静默忽略，需要恢复。

实现拆解

实现按以下 5 步拆解：

1. 核心重写为类式 rollout 函数（miles/rollout/fully_async_rollout.py，+209/-304）：删除 AsyncRolloutWorker、get_global_worker/stop_global_worker 全局单例、线程与 atexit，新增 FullyAsyncRolloutFn。构造函数接收 RolloutFnConstructorInput，持有实例级 GenerateState、dynamic/sample filter、_CachedWeightVersion；__call__首次训练调用时懒启动 _worker 任务与 asyncio.Queue(maxsize=1000)；_worker_loop 生产者维持 in-flight 上限（rollout_batch_size 或 async_max_concurrent_samples 折算的组数），队列满时 put 阻塞形成背压；_drain 消费者回收 ABORTED/ 超龄分组（注意回收 prompt_group 而非生成结果，修复了旧实现多采样组 abort 回收路径的隐式 AttributeError），应用 dynamic filter（丢弃而非回收）与 sample filter（标记 remove_sample 不缩 batch），最终按 index 排序；新增 assert len(group) == n_samples_per_prompt 与 per-step metrics（queue_size、recycle 计数、staleness 统

计）。

2. 参数解析收敛到单一决策点（miles/utils/arguments.py）：--rollout-function-path 默认值从计算出的字符串改为 None，新增 --fully-async flag；resolve_rollout_function_paths 成为唯一回答 " 参数选择了哪个 rollout/eval 函数 " 的入口，且 eval 在 fully-async override 之前解析（"fully async 不服务 eval" 由构造保证）；_resolve_rollout_functions 在 miles_validate_args 中串联一组互斥断言（需要 refactor 环境变量、不与 multi-LoRA/colocate/partial-rollout/recompute-logprobs-via-prefill/rollout-all-samples-process 冲突）；train.py 追加一行直接拒绝 --fully-async。miles/utils/multi_lora.py 同步把 " 用户未自选 rollout fn" 的判断从字符串比较改为 is-None（旧比较在默认值变为 None 后静默失效，会让 multi-LoRA 跑回默认 rollout fn）。
3. 错误处理与细节修复：_CachedWeightVersion.get 的 _last_query 改为 finally 打戳（失败也被节流，否则不可达 router 让每个 group 都付满 2s 超时），去掉 _value is not None 短路；_next_group 先检查 worker 任务再取队列，死亡的 worker 立即失败 step 而非先 drain 积压旧数据（review 发现的问题，commit 2d47501 修复）；提取 _iter_samples/_first_sample 统一嵌套 Group 解包。
4. 示例与文档迁移：examples/fully_async/run_qwen3_30b_a3b_fully_async.py、run-qwen3-4b-fully_async.sh、examples/swe-agent/run-glm47-flash-agentic-async.py 从 --rollout-function-path 改为 --fully-async 并补 MILES_EXPERIMENTAL_ROLLOUT_REFACTOR=1；docs/examples/fully-async.md 重写 walkthrough（含 mermaid 时序图），docs/user-guide/fully-async.md、training-script-walkthrough.md、architecture.md 同步；examples/infra_features/random_async/random_async_rollout.py 的 docstring 引用更新。
5. 测试配套：tests/fast/rollout/test_fully_async_rollout.py（新增，12 个 stage-a-cpu 用例，FakeGenerateState/FakeDataSource 桩精确刻画状态机，覆盖冷启动、回收、死 worker、in-flight 上限、动态过滤不回收、权重版本失败节流）；tests/e2e/megatron/test_qwen3_30B_A3B/test_fully_async.py（新增，stage-c-8-gpu-h100，disaggregated 拓扑 colocate=False，3 次 rollout 覆盖冷启动 / 暖队列 drain/ 跨权重更新回收，注册 5 个 CI gate 指标）；_common.py 的 CaseConfig 增加 fully_async/num_rollout 字段并在配置阶段拒绝 colocate；tests/ci/labels.py 注册 fully-async 标签。

关键文件：

- miles/rollout/fully_async_rollout.py（模块 异步引擎；类别 source；类型 core-logic；符号 FullyAsyncRolloutFn, AsyncRolloutWorker, get_global_worker, stop_global_worker）：核心重写载体：删除线程 / 全局单例 /atexit/print，FullyAsyncRolloutFn 以共享事件循环上的长生命任务实现生产者 - 消费者 + 背压；错误显式化、回收用 prompt_group、weight-version 失败节流等行为修复都集中在此。
- tests/fast/rollout/test_fully_async_rollout.py（模块 单元测试；类别 test；类型 test-coverage；符号 FakeGenerateState, FakeDataSource, make_group, make_args）：12 个 CPU 用例用 FakeGenerateState/FakeDataSource 桩精确刻画状态机：回收、死 worker 传播、in-flight 上限、动态过滤不回收、权重版本失败节流、worker 跨调用持久化

等。

- miles/utils/arguments.py（模块 参数解析；类别 source；类型 core-logic；符号 resolve_rollout_function_paths, _resolve_rollout_functions）：参数选择收敛到 resolve_rollout_function_paths 单一决策点，--fully-async 与互斥断言在此定义；eval 先于 override 解析，从构造上保证 fully-async 不服务 eval。
- tests/e2e/megatron/test_qwen3_30B_A3B/test_fully_async.py（模块 端到端测试；类别 test；类型 test-coverage）：8×H100 disaggregated 拓扑 e2e：3 次 rollout 覆盖冷启动、暖队列 drain、跨权重更新回收，注册 5 个 CI gate 指标，验证持久 worker 才存在的状态。
- tests/e2e/megatron/test_qwen3_30B_A3B/_common.py（模块 测试基建；类别 test；类型 test-coverage；符号 CaseConfig）：CaseConfig 增加 fully_async/num_rollout 字段，按开关选择 train_async.py 并在配置阶段拒绝 colocate，是 e2e 基建的关键配套。
- miles/utils/multi_lora.py（模块 多 LoRA；类别 source；类型 core-logic；符号 validate_multi_lora_args）：适配 rollout_function_path 默认值改为 None：is-None 判断取代与标准路径的字符串比较，否则 multi-LoRA 会静默跑回默认 rollout fn。
- examples/fully_async/run_qwen3_30b_a3b_fully_async.py（模块 示例脚本；类别 source；类型 core-logic）：示例迁移到正式 --fully-async 入口并补充 MILES_EXPERIMENTAL_ROLLOUT_REFACTOR=1 环境变量，是脚本 / 文档迁移的代表。
- docs/examples/fully-async.md（模块 文档；类别 docs；类型 documentation；符号 FullyAsyncRolloutFn, _worker_loop, _drain）：walkthrough 从全局 worker 模式改写为类 API 设计，明确记录背压、错误显式化、无 eval 限制等关键语义，是理解设计意图的最佳文档。

关键符号：FullyAsyncRolloutFn.call, FullyAsyncRolloutFn._worker_loop, FullyAsyncRolloutFn._next_group, FullyAsyncRolloutFn._drain, FullyAsyncRolloutFn._recycle, FullyAsyncRolloutFn._submit_one_group, _CachedWeightVersion.get, resolve_rollout_function_paths, _resolve_rollout_functions, _iter_samples, _first_sample

关键源码片段

miles/utils/arguments.py

参数选择收敛到 resolve_rollout_function_paths 单一决策点，--fully-async 与互斥断言在此定义；eval 先于 override 解析，从构造上保证 fully-async 不服务 eval。

```
# resolve_rollout_function_paths：把「参数选择了哪个 rollout / eval 函数」
# 收敛到唯一入口。--rollout-function-path 默认改为 None 后，" 用户是否自己
# 指定过 " 就是一次 is-None 判断，不再与计算出的默认值做字符串比较。

def resolve_rollout_function_paths(args) -> tuple[str, str]:
    if enable_experimental_rollout_refactor():
        standard_path = 'miles.rollout.inference_rollout.inference_rollout_common.
        InferenceRolloutFn'
    else:
        standard_path = 'miles.rollout.sglang_rollout.generate_rollout'
    rollout_path = args.rollout_function_path or standard_path
```

```
    # eval 在 fully-async override 之前解析："fully async 不服务 eval"
    # 因此由构造保证 —— eval 总是标准路径，除非用户显式指定
    eval_path = args.eval_function_path or rollout_path
    if args.fully_async:
        rollout_path = 'miles.rollout.fully_async_rollout.FullyAsyncRolloutFn'
    return rollout_path, eval_path


def _resolve_rollout_functions(args) -> None:
    # 一组 fail-fast 断言：与其让某些 flag 被静默忽略（旧实现正是如此），
    # 不如在参数校验阶段直接拒绝无法正确工作的组合
    if args.fully_async:
        assert enable_experimental_rollout_refactor(), (
            '--fully-async needs the class-based rollout API: '
            'set MILES_EXPERIMENTAL_ROLLOUT_REFACTOR=1'
        )
        # 注意 multi-LoRA 会在本函数之前自行选择 rollout 函数
        assert not args.multi_lora, '--fully-async 与 multi-LoRA 选择不同的 rollout 函数'
        assert args.rollout_function_path is None, '--fully-async 与 --rollout-function-path 互斥'
        assert not args.colocate, '--fully-async 无法 colocate：rollout 需在训练期间持续生成'
        assert not args.partial_rollout, '--fully-async 不支持 --partial-rollout'
        assert (
            not args.recompute_logprobs_via_prefill
        ), '--fully-async 不支持 --recompute-logprobs-via-prefill'
        assert (
            args.rollout_all_samples_process_path is None
        ), '--fully-async 不支持 --rollout-all-samples-process-path'

    args.rollout_function_path, args.eval_function_path = resolve_rollout_function_paths(args)
```

评论区精华

Review 中 gemini-code-assist[bot] 提出 4 条意见：

- high：RolloutManager 共享 train/eval 实例的并发风险（miles/ray/rollout/rollout_manager.py）：eval_function_path == rollout_function_path 时复用 generate_rollout 实例，而 InferenceRolloutFn 的 GenerateState 含可变状态（aborted、semaphore），训练被打断时并发 eval 会被静默中止。作者有意复用（避免 FullyAsyncRolloutFn 被构造两次、保持状态），guapisolo 在 Issue 评论中确认接受现状："CC said eval has some conflict with current fully async code path. But I think we can leave this PR as-is." 最终 APPROVED："LGTM. clean design."。
- high：_CachedWeightVersion 解析在 try-except 外：int(data['weight_version']) 若遇畸形响应会抛 KeyError/ValueError 使训练进程崩溃，建议一并捕获；head 版本未采纳，仍是潜在 crash 点。
- medium：_next_group 的 queue_get 任务泄漏：建议 try-finally 取消；已被采纳，head 版本含 finally cancel 且 worker 检查在队列之前（commit 2d47501）。

- medium：assert 校验应改 ValueError：断言在 python -O 下会被全局禁用；未采纳，仓库现有校验风格（如 _common.py）同样大量使用 assert。

guapisolo 另补了两个小提交：docs 补 MILES_EXPERIMENTAL_ROLLOUT_REFACTOR=1（否则文档中的启动命令按 --fully-async 硬断言会启动失败）与提取 _first_sample helper。

- RolloutManager 共享 train/eval 实例的并发风险 (correctness): 作者有意复用（避免 FullyAsyncRolloutFn 构造两次并保持状态），guapisolo 在 Issue 评论中确认接受现状，最终 APPROVED。
- _CachedWeightVersion 解析在 try-except 外 (correctness): head 版本未采纳，仍是潜在 hard crash 点。
- _next_group 的 queue_get 任务泄漏 (correctness): 已修复：head 版本含 finally cancel，且 worker 检查先于队列检查。
- assert 校验应改用 ValueError (style): 未采纳，仓库现有校验风格（如 _common.py、_resolve_rollout_functions）大量使用 assert，保持一致。
- eval 与 fully-async 代码路径的冲突 (design): 接受现状：eval 由解析顺序保证走标准 InferenceRolloutFn，fully-async 显式 raise。

风险与影响

- 风险：
  1. _CachedWeightVersion.get 未完全加固（miles/rollout/fully_async_rollout.py）：int(data['weight_version']) 和 data['weight_version'] 在 except 之外，router 返回畸形 JSON 时 KeyError/ValueError 会直接传出，导致训练进程崩溃——而 staleness filter 本应 best-effort（注释明言）。这是 review 未采纳建议，风险实存。
  2. 共享实例的并发隐患：RolloutManager 在 eval_function_path == rollout_function_path 时复用实例，影响所有类 API 用户（不止 fully-async）；GenerateState 的 aborted/semaphore 若在 train/eval 并发时互相污染，会产生难排查的静默行为。本 PR 未改动 rollout_manager 的状态隔离。
  3. fail-fast 带来的行为断裂：--fully-async 与 --partial-rollout、--recompute-logprobs-via-prefill、--rollout-all-samples-process-path、colocate、multi-LoRA 互斥。旧实现静默忽略这些 flag，新实现启动即 assert 失败；若现有脚本组合了这些参数，升级后立即失败（刻意设计，但对使用方是破坏性变更）。
  4. 单事件循环共享：worker 与 _drain 在同一 loop，_drain 内任何阻塞（如 weight-version 查询 2s 超时）都会暂停 worker 提交，吞吐受消费侧影响；这是单 loop 模型的固有权衡。
  5. 配置校验依赖 assert：_resolve_rollout_functions 与 _drain 的断言在 python -O 下全部失效（gemini 指出），关键互斥校验绕不过时静默降级。 - 影响：对使用者：fully-async 从 " 手动填 --rollout-function-path + 依赖旧堆栈 " 变为正式 flag，但要求 MILES_EXPERIMENTAL_ROLLOUT_REFACTOR=1、train_async.py 专属、colocate=False（不能与训练共享 GPU）；train.py 直接拒绝。对系统：per-iteration wall time 从 rollout_time + train_time 趋向 max(rollout_time, train_time)；worker 生命周期与 rollout fn 实例绑定，进程内不再有全局单例；错误不再被静默吞掉。对团队与

后续演进：_submit_one_group 已为 #1673 sample-completion backfill 预留形状；examples/swe-agent 的 run-glm47-flash-agentic-async.py 因此真正获得 dynamicfilter 支持；rollout 类 API 生态（#1916/#1759 重构序列）再进一步。测试投入大：12 个 CPU 用例 + 8xH100 e2e（约 1500s），CI 新增 fully-async 标签可单独触发。- 风险标记：核心路径重写，fail-fast 行为破坏，weight_version 解析未加固，共享实例并发隐患，assert 校验可被 -O 绕过

关联脉络

- PR #1716 Move fully-async rollout from examples into miles/rollout: 本 PR 的直接前驱：把 example 质量的实现移入核心库，本 PR 在此基础上重写为类 API（PR body 明确引用 #1716）。
- PR #1916 (1/2) refactor(rollout): drop --generate-multi-samples and its per-turn sample semantics: 同一 rollout 类 API 重构序列的前半部分，多轮轨迹统一返回标量 Sample，决定了本 PR 中 Group 嵌套形状与 n_samples_per_prompt 断言的处理。
- PR #1759 (2/2) refactor(session): assemble training samples on the session server; records never leave it: 同一重构序列的后半部分，训练样本组装与数据链路改造，与 rollout 类 API 的边界划分相关。
- PR #1735 [PPO] Share Actor/Critic GPUs: train_async 路径的 colocate 语义相关：fully-async 禁止 colocate，与共享 GPU 拓扑的选择相互约束。
- PR #1829 fix: require explicit off-policy correction for async PPO training: 同为 train_async.py 入口的参数校验强化，与 --fully-async 的 fail-fast 断言风格一致。