PR #2244 完整报告

radixark/miles

pass the engine weight version from the trainer instead of polling the router

合并时间：2026-08-08 05:56

原文链接：http://prhub.com.cn/radixark/miles/pull/2244

执行摘要

- 一句话：权重版本由训练器直传，去掉 router 轮询
- 推荐动作：值得精读。核心设计决策是“把权威状态从轮询改为推送”：引擎版本号本来就是训练器写出的，读回它只会引入延迟和抖动。值得关注的点：打版本戳统一上移到 mixin._finalize_and_resume_engines 以覆盖 LoRA，以及 set_weight_version 对版本回退的 assert/ 告警二态处理。建议跟进两个后续：--indep-dp 下版本计数器的持久化 / 恢复（P2），以及补 LoRA 路径的 staleness 测试。

功能与动机

PR body 指出：FullyAsyncRolloutFn 需要当前引擎权重版本衡量并限制 sample 陈旧度，但旧实现是每次 drain group 后通过 HTTP 轮询 router /model_info（带 1s TTL），而这个数字本来就是训练器自己写进引擎的（update_weight_version(weight_version=str(self.weight_version)）），轮询只是读回刚写下的值。改成直传后：不再因 router 抖动降级为 None，也不再需要特判引擎更新前的 "default" 字符串，并且为 #2030 把 staleness 控制搬进 fully-async 数据缓冲扫清了每次 HTTP 往返的成本。

实现拆解

实现按 5 步拆解：

1. 数据契约扩展：miles/rollout/base_types.py 给 RolloutFnTrainInput 增加 weight_version: int | None = None 字段，镜像已有的 RolloutFnEvalInput。冻结 dataclass 新增带默认值字段不会破坏既有按位置传参的调用点。
2. 训练侧推送：miles/backends/megatron_utils/actor.py 与 miles/backends/experimental/fsdp_utils/actor.py 都在 rank 0 于 update_weights() 返回后调用 self.rollout_manager.set_weight_version.remote(self.weight_updater.weight_version)，与既有 clear_updatable_has_new_engines 调用同一前置条件（权重推送已结束）。
3. RolloutManager 承接：miles/ray/rollout/rollout_manager.py 新增 weight_version 属性和 set_weight_version 方法；方法内对版本回退做检查——--indep-dp 容错模式下仅 logger.warning，否则直接 assert 失败；_get_rollout_data 构造 RolloutFnTrainInput 时带上 weight_version=self.weight_version。

4. 消费侧去轮询：miles/rollout/fully_async_rollout.py 删除 _CachedWeightVersion 类、httpx/time 依赖、WEIGHT_VERSION_QUERY_TIMEOUT_SECS 常量；_drain 签名从 rollout_id: int 改为 input: RolloutFnTrainInput，staleness 计算直接使用 input.weight_version。
5. 引擎打戳统一：把 engine.update_weight_version.remote(...) 从 p2p 传输路径的 _finalize_and_resume_engines 重写中上移到 mixin.py 的公共 _finalize_and_resume_engines，覆盖 full-param 与 LoRA 两类更新路径（对应 commit a4cb8d3）。

配套测试：tests/fast/rollout/test_fully_async_rollout.py 中 test_stale_group_recycled 改为直接 RolloutFnTrainInput(weight_version=10)；删除 test_weight_version_throttles_failed_queries，新增 test_staleness_filter_off_before_the_first_weight_update 验证首轮更新前 None 语义（过滤关闭而非 staleness=0）。

关键文件：

- miles/rollout/fully_async_rollout.py（模块 异步驱动；类别 source；类型 dependency-wiring；符号 _CachedWeightVersion, _drain）：核心消费路径：删除 _CachedWeightVersion 轮询器，_drain 改为从 RolloutFnTrainInput.weight_version 读取当前版本，是本次重构的主战场（+4/-35）。
- miles/ray/rollout/rollout_manager.py（模块 管理器；类别 source；类型 core-logic；符号 set_weight_version）：新增 weight_version 状态与 set_weight_version 推送入口，并在 _get_rollout_data 中把版本注入 RolloutFnTrainInput，是训练器到 rollout 的衔接点，也承载版本回退检查。
- miles/backends/megatron_utils/update_weight/update_weight_from_distributed/mixin.py（模块 权重更新；类别 source；类型 core-logic；符号 _finalize_and_resume_engines）：把引擎打版本戳统一到公共 _finalize_and_resume_engines，同时覆盖 full-param 与 LoRA 更新路径，直接回应 reviewer 的 P1 评论。
- miles/backends/megatron_utils/update_weight/update_weight_from_distributed/p2p.py（模块 权重更新；类别 source；类型 core-logic；符号 _finalize_and_resume_engines）：删除该路径自己的 _finalize_and_resume_engines 重写，避免与 mixin 公共实现重复打戳。
- miles/rollout/base_types.py（模块 基类型；类别 source；类型 data-contract；符号 RolloutFnTrainInput）：RolloutFnTrainInput 增加 weight_version 字段，是训练器到 fully-async rollout 的数据契约变更。
- miles/backends/megatron_utils/actor.py（模块 Megatron 后端；类别 source；类型 core-logic；符号 update_weights）：Megatron 训练 actor 在 update_weights 后 rank 0 推送版本号到 RolloutManager。
- miles/backends/experimental/fsdp_utils/actor.py（模块 FSDP 后端；类别 source；类型 core-logic；符号 update_weights）：FSDP 训练 actor 同步接入推送逻辑，保持两个后端的 fully-async 行为一致。
- tests/fast/rollout/test_fully_async_rollout.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_stale_group_recycled, test_staleness_filter_off_before_the_first_weight_update）：测试从

FakeWeightVersion 注入改为 input 传参，新增 None 语义覆盖，删除轮询节流测试，是本次行为变化的主要验证。

关键符号：FullyAsyncRolloutFn._drain, RolloutManager.set_weight_version, DistBucketedWeightUpdateMixin._finalize_and_resume_engines, MegatronActor.update_weights, FSDPActor.update_weights

关键源码片段

miles/rollout/fully_async_rollout.py

核心消费路径：删除 _CachedWeightVersion 轮询器，_drain 改为从 RolloutFnTrainInput.weight_version 读取当前版本，是本次重构的主战场（+4/-35）。

```
async def _drain(self, input: RolloutFnTrainInput) -> RolloutFnTrainOutput:
    args = self.args
    assert args.rollout_global_dataset

    target_data_size = args.rollout_batch_size
    data: list[Group] = []
    aborted_groups_recycled = 0
    stale_groups_recycled = 0
    staleness_values: list[int] = []
    metric_gatherer = MetricGatherer()
    do_print = True

    while len(data) < target_data_size:
        prompt_group, group = await self._next_group()
        assert len(group) == args.n_samples_per_prompt

        # 权重更新打断了生成：整组退回数据源重新采样
        if any(s.status == Sample.Status.ABORTED for s in _iter_samples(group)):
            self._recycle(prompt_group)
            aborted_groups_recycled += 1
            continue

        if args.max_weight_staleness is not None:
            oldest = group_oldest_weight_version(group)
            # 引擎权重版本现在由训练器经 RolloutFnTrainInput 直传，
            # 不再每消费一个 group 就轮询 router 的 /model_info
            current = input.weight_version
            if oldest is not None and current is not None:
                staleness = current - oldest
                staleness_values.append(staleness)
                if staleness > args.max_weight_staleness:
                    self._recycle(prompt_group)
                    stale_groups_recycled += 1
                    logger.info(
                        f"Recycled stale group (oldest_version={oldest}, current={current}, "
                        f"staleness={staleness} > max={args.max_weight_staleness})"
```

```
                )
                continue

        # 动态过滤：被丢弃的组不进训练也不回收
        filter_output = call_dynamic_filter(self._dynamic_filter, args, group)
        if not filter_output.keep:
            metric_gatherer.on_dynamic_filter_drop(reason=filter_output.reason)
            continue

        data.append(group)
    # 后续：首样本日志、指标汇总与 RolloutFnTrainOutput 组装
```

miles/ray/rollout/rollout_manager.py

新增 weight_version 状态与 set_weight_version 推送入口，并在 _get_rollout_data 中把版本注入 RolloutFnTrainInput，是训练器到 rollout 的衔接点，也承载版本回退检查。

```
@ray.remote
class RolloutManager:
    def __init__(self, args, pg):
        self.pg = pg
        self.args = args
        # 由训练 actor 在每次权重更新后写入；None 表示尚未发生权重更新
        self.weight_version: int | None = None

    def set_weight_version(self, weight_version: int):
        # 版本回退说明另一个 cell 接管了更新（--indep-dp 容错场景），
        # 此时只告警；否则直接断言失败以暴露计数器不一致
        if self.weight_version is not None and weight_version < self.weight_version:
            message = f"Engine weight version went backwards: {self.weight_version} -> {weight_
            version}"
            assert self.args.indep_dp, message
            logger.warning(message)
        self.weight_version = weight_version
```

miles/backends/megatron_utils/update_weight/update_weight_from_distributed/mixin.py

把引擎打版本戳统一到公共 _finalize_and_resume_engines，同时覆盖 full-param 与 LoRA 更新路径，直接回应 reviewer 的 P1 评论。

```
def _finalize_and_resume_engines(self) -> None:
    """关闭权重更新会话并恢复 rollout 引擎。"""
    if dist.get_rank() == 0:
        # 统一在此处打版本戳：覆盖 full-param 与 LoRA 两条路径。
        # 若只在 p2p 传输路径打戳，LoRA 加载后 sample.weight_versions
        # 仍停留在旧值，staleness 过滤会失去参照
        ray.get(
            [
                engine.update_weight_version.remote(weight_version=str(self.weight_version))
                for engine in self.rollout_engines
```

```
            ]
        )
        end_weight_update(self.rollout_engines)
        ray.get([engine.continue_generation.remote() for engine in self.rollout_engines])
```

评论区精华

核心讨论集中在两个问题上：

1. LoRA 路径的版本戳缺失（P1，guapisolo）：reviewer 指出 input.weight_version 只有在与 Sample.weight_versions 同源时才可比；而单 LoRA + 默认 broadcast 传输路径 load_lora_adapter_from_distributed 不打任何 weight_version，样本会残留 "default" 或固定基础版本，导致 oldest_weight_version 要么返回 None 静默关闭过滤，要么固定不变直到整组被当成 stale 回收。作者通过把打戳统一进 mixin._finalize_and_resume_engines 解决（commit 信息即 “stamp the engine weight version for every transport including LoRA”），但未按 reviewer 要求补 LoRA 专项测试。
2. 容错切换下版本回退（P2，guapisolo 的 issue 评论）：weight_updater.weight_version 是每个 actor 的内存计数器，不 checkpoint、不持久化；--indep-dp 下 update_weights 跑在第一个存活的 cell 上，cell 0 死后 cell 1 的计数器仍为 0，版本会从 57 回退到 1，已入队的旧版本样本使 staleness = current - oldest 变负，反而恰好把最 off-policy 的组放进训练。当前实现（--indep-dp 只告警）没有闭环该问题。

- LoRA 路径未打版本戳导致 staleness 参照失效 (correctness): 作者将引擎打戳统一上移到 mixin._finalize_and_resume_engines，覆盖 LoRA 与 full-param 路径（commit a4cb8d3）；但 reviewer 要求的 LoRA 专项测试未补充。
- cell failover 下 weight_version 回退使 staleness 过滤静默失效 (design): set_weight_version 在 --indep-dp 下只告警不阻断，且 RolloutManager.save/load 不持久化版本号，P2 隐患未闭环。

风险与影响

- 风险：
  1. 容错切换版本回退（P2 未闭环）：set_weight_version 在 --indep-dp 下仅 logger.warning，版本回退后 _drain 中 staleness 变负、过滤静默失效，rollout/fully_async/{avg,max}_staleness 指标也会出现负数。需要后续把版本计数纳入 checkpoint 或从引擎恢复。
  2. LoRA 路径缺专项测试：虽然打戳已统一到 mixin._finalize_and_resume_engines，若未来有 LoRA 传输路径绕过该 hook，P1 问题会复发；reviewer 明确要求补覆盖。
  3. 通知时序窗口：set_weight_version 在 update_weights() 返回（引擎已恢复生成）之后才到达，期间 drain 会少报 staleness；PR body 自述该窗口小于被替换的 1s TTL，但仍存在。
  4. 首轮语义变化：第一个权重更新前 weight_version 为 None，staleness 过滤显式关闭；这是有意为之，但若用户在预热阶段依赖过滤会误以为配置失效。 - 影响：影响集中在 fully-async rollout 与权重更新链路：删除每 group 一次的 router HTTP 查询（含失败 2s 超时风险），drain 延迟更稳定；RolloutManager 与训练 actor 新增接口，

Megatron 与 FSDP 两个后端同步改动；RolloutFnTrainInput 契约扩展影响所有实验性 rollout 实现。对用户而言，staleness 过滤在首轮权重更新前被显式关闭、router 抖动不再降级为 None；对团队而言，这是为 #2030 把 staleness 控制下沉到数据缓冲区的前置清理。 - 风险标记：核心 rollout 路径变更，容错切换版本回退未闭环，LoRA 路径缺专项测试，过滤静默关闭语义变化

关联脉络

- PR #2030 staleness control in fully-async data buffer (PR body 提及): PR body 明确说明本 PR 是 #2030 的前置：把 staleness 控制下沉到 fully-async 数据缓冲区并支付每次 HTTP 往返成本，本 PR 先消除该成本，独立合入 main。
- PR #1673 [feat] support sample rollout submission granularity to keep fully async concurrency: 同文件 miles/rollout/fully_async_rollout.py 的 fully-async 并发与回收机制持续演进，本 PR 在此基础上切换了 weight_version 的来源。
- PR #2235 feat(fsdp): support rollout routing replay (R3) for fsdp backend: 同样改动 miles/backends/experimental/fsdp_utils/actor.py，FSDP 后端的 rollout/ 权重更新集成是本 PR 需要保持对齐的上下文。