

PR #2824 完整报告

radixark/miles

Temporarily revert PR 1313

合并时间: 2026-09-04 12:04

原文链接: <http://prhub.com.cn/radixark/miles/pull/2824>

执行摘要

- 一句话: 临时回滚 RDT 权重同步功能, 为后续在重构架构上重新引入做准备。
- 推荐动作: 此 PR 是一次有计划的、服务于更大重构目标的临时性破坏。强烈建议关注后续重构 PR (如 #1842, #1843) 的进展, 以评估 RDT 功能何时以及如何被恢复。对于正在使用 RDT 的用户, 需要提前预警并规划迁移路径。对于评估系统架构的读者, 此 PR 揭示了项目在演进过程中处理复杂功能模块的策略: 先安全回滚, 再在新基础上重建。

功能与动机

根据 commit message 和 PR body, 此变更属于 #1837 重构跟踪 issue 的一部分。commit message 明确指出是为了 'rebased without carrying the server_group / actor_factory plumbing that the chain removes', 即简化重构链的 rebase 过程, 移除将在后续重构中被清除的旧有代码。PR body 关联了 #1837, 而该 issue 是追踪一系列重构和增强工作的主 issue。review 评论中作者 fzyzcjy 也指出: 'The op removes an existing RDT weight-sync mode, but the designated safe deliver still does not restore it', 确认了这是为重构链的演进所做的临时性清理。

实现拆解

1. 回滚 RDT 核心实现: 删除了 miles/backends/training_utils/weight_update/protocols/rdt.py 文件, 其中包含了 UpdateWeightFromRDT 协议类、_EngineRankBucket GPU 桶管理类以及所有与 NIXL RDMA 传输相关的逻辑。同时删除了 miles/ray/rollout/sglang_server_actor.py, 移除了负责在 Ray 中托管 SGLang HTTP 服务器的 SGLangServerActor, 这是 RDT use_ray=True 模式的核心进程。
2. 清理 SGLang 引擎集成: 在 miles/backends/sglang_utils/sglang_engine.py 中, 移除了 _launch_sglang_server 函数 (该函数创建并启动 SGLangServerActor) 以及 SGLangEngine 类中的 RDT 特定初始化逻辑 (如设置 use_ray/enable_rdt_weight_sync 参数、处理多节点启动) 和 get_scheduler_actors 方法。shutdown 方法也相应简化, 移除了对 RDT actor 的处理。
3. 简化资源分配: 在 miles/ray/placement_group.py 中, 移除了 _create_placement_group 函数中的 is_rdt 参数和专为 RDT 设计的 placement group bundle 排序逻辑 (该逻辑确保 trainer 节点被填满, 为 rollout 留出空间)。allocate_train_group 中 RDT 模式下 num_gpus_per_actor=1 的特殊分配被移除, 统一为 0.4。miles/ray/rollout/server_group.py 中启动引擎时, 移除了 RDT 模式下复用

placement group 和传递 pg_bundles 的逻辑。miles/ray/train/actor_factory.py 中移除了 RDT 特定的环境变量设置（如 RAY_EXPERIMENTAL_NOSET_CUDA_VISIBLE_DEVICES）和并发度调整。

4. 清理参数验证和协议注册：在 miles/utils/arguments.py 中，从 --update-weight-transfer-mode 参数的选项中移除了 'rdt'，并删除了相关的验证断言（如要求 Megatron 后端、不兼容 PPO 等）。在 miles/backends/training_utils/weight_update/protocol.py 的 get_weight_transfer_protocol 函数中，移除了根据 rdt 模式创建 UpdateWeightFromRDT 的分支。
5. 移除测试：删除了 RDT 专用的端到端测试 tests/e2e/megatron/test_qwen3_30B_A3B_rdt.py，并从 tests/fast/utils/test_arguments.py 中移除了 TestRdtValidation 测试类。其他测试文件（如 conftest.py, test_actor_factory.py）也移除了与 RDT 相关的 mock 或条件。

关键文件：

- miles/backends/training_utils/weight_update/protocols/rdt.py（模块 权重更新协议；类别 source；类型 deletion；符号 _staging_span, _EngineRankBucket, init, stage）：RDT 权重同步协议的核心实现，整个文件被删除，代表被移除的核心功能。
- miles/backends/training_utils/weight_update/protocol.py（模块 权重更新协议；类别 source；类型 dependency-wiring）：权重传输协议的工厂函数，移除了 RDT 协议的注册分支，是功能移除的关键接入点。
- miles/utils/arguments.py（模块 参数定义；类别 source；类型 configuration）：命令行参数定义与验证，移除了 rdt 选项和相关的约束断言，从用户接口层禁用了此功能。
- miles/backends/sglang_utils/sglang_engine.py（模块 SGLang 引擎；类别 source；类型 core-logic；符号 _launch_sglang_server, get_scheduler_actors）：SGLang 推理引擎门面，移除了 RDT 模式下的启动、健康检查和关闭逻辑，是 RDT 运行时集成的关键文件。
- tests/e2e/megatron/test_qwen3_30B_A3B_rdt.py（模块 RDT 测试；类别 test；类型 deletion；符号 prepare, execute）：RDT 模式的端到端测试，整个文件被删除，移除了对 RDT 功能的集成测试覆盖。

关键符号：UpdateWeightFromRDT.connect, UpdateWeightFromRDT.create_gpu_replica, get_weight_transfer_protocol, SGLangEngine._init_normal, _launch_sglang_server, _create_placement_group, miles_validate_args

关键源码片段

miles/backends/training_utils/weight_update/protocols/rdt.py

RDT 权重同步协议的核心实现，整个文件被删除，代表被移除的核心功能。

```
# 被删除的 `UpdateWeightFromRDT.connect` 方法展示了 RDT 如何建立传输连接：
# 1. 根据 `RemoteTransferPlan` 计划 P2P 传输目标。
# 2. 为每个目标引擎 rank 创建一个 GPU 模型副本（`create_gpu_replica`）和一个固定大小的 GPU 桶。
# 3. 从 SGLang 获取 SchedulerActor 句柄，并注册 NIXL 内存传输。
# 整个方法体现了基于 NIXL 的 RDMA 拉取传输的初始化过程。
def connect(self, rollout_engines, ...):
    targets = self.transfer_plan.plan_p2p()
```

```

for engine_rank, rank_targets in targets_grouped_by_engine_rank.items():
    model_replica, params_dict, param_specs = self.create_gpu_replica(...)
    gpu_bucket = torch.empty(self.args.update_weight_buffer_size, dtype=torch.uint8, device=
    ...)
    register_nixl_memory(gpu_bucket) # 关键：注册桶内存用于 RDMA 传输
    actors = [self._get_engine_scheduler_actors(rollout_engines, t.engine_ind)[t.engine_rank]
    for t in rank_targets]
    self._engine_rank_buckets.append(
        _EngineRankBucket(model_replica, params_dict, param_specs, gpu_bucket, actors)
    )

```

miles/backends/training_utils/weight_update/protocol.py

权重传输协议的工厂函数，移除了 RDT 协议的注册分支，是功能移除的关键接入点。

修改后的协议选择逻辑，`rdt` 分支已被移除。

```

def get_weight_transfer_protocol(args: Namespace) -> WeightTransferProtocol:
    if args.colocate:
        from miles.backends.training_utils.weight_update.protocols.cuda_ipc import
        UpdateWeightFromTensor
        return UpdateWeightFromTensor(args)
    # ... 其他模式 ...
    # if args.update_weight_transfer_mode == "rdt": # <-- 此分支被移除
    # from miles.backends.training_utils.weight_update.protocols.rdt import
    # UpdateWeightFromRDT
    # return UpdateWeightFromRDT(args)
    if args.update_weight_transfer_mode == "p2p":
        from miles.backends.training_utils.weight_update.protocols.p2p import UpdateWeightP2P
        return UpdateWeightP2P(args)

```

miles/backends/sglang_utils/sglang_engine.py

SGLang 推理引擎门面，移除了 RDT 模式下的启动、健康检查和关闭逻辑，是 RDT 运行时集成的关键文件。

简化后的 `SGLangEngine.\_init\_normal` 方法，移除了所有 RDT 相关分支。

```

def _init_normal(self, server_args_dict):
    # 旧代码：
    # use_rdt = self.args.update_weight_transfer_mode == "rdt"
    # if use_rdt:
    # ...
    # self._sglang_server_actor, self._scheduler_actors = _launch_sglang_server(server_args, ...)
    # else:
    # self.process = launch_server_process(server_args)

    # 新代码：统一使用进程模式启动
    server_args = ServerArgs(**{**server_args_dict, "host": server_args_dict["host"].strip("[]")})
    self.process = launch_server_process(server_args)

```

评论区精华

- 关键决策与风险: fzyzcjy 在 miles/utils/arguments.py 的 review 评论中明确指出了此次回滚带来的风险: '[P1] Restore RDT before deliver-1 Existing launchers that pass --update-weight-transfer-mode rdt now fail during argument parsing because this choice and the supporting backend were removed.' 这确认了回滚的临时性和当前版本的功能性回退。
- 重构意图明确: commit message 和 PR body 都清晰地将此次回滚定位为服务于更大的重构链 (#1837)。目标是移除需要被重写的“plumbing” (布线代码), 以便在新架构上重新实现 RDT。这反映了在大型重构中, 有时需要先进行破坏性清理, 再构建新实现的工程策略。
- 无其他未解决疑虑: 除作者自身的评论外, 没有其他 reviewer 提出反对意见或技术疑问。
 - RDT 功能移除的风险与恢复计划 (correctness): 确认回滚是临时性的, 为重构铺路。RDT 功能将在后续基于重构后架构的 PR 中重新实现。当前版本存在已知的功能性回退。

风险与影响

- 风险:
 1. 功能回退 (核心风险): 完全移除了 rdt 权重更新模式。所有依赖 --update-weight-transfer-mode rdt 的现有配置和脚本将无法运行, 在参数解析阶段即会失败。此功能的恢复需等待后续 PR 基于重构后的结构重新实现。
 2. 兼容性破坏: 破坏了向后兼容性。下游用户如果使用了 RDT 模式, 升级到包含此 PR 的版本后, 其训练流程会立即中断。
 3. 测试覆盖缺口: 删除了对 RDT 功能的主要端到端测试和单元测试覆盖。虽然这是预期的回滚, 但在 RDT 重新实现之前, 该功能路径将没有自动化测试保护。
 4. 重构依赖: 此次回滚的完整价值 (为重构铺路) 依赖于后续 PR (如 #1842, #1843 等) 的成功实施。如果重构链停滞, RDT 功能的缺失可能成为长期问题。
- 影响:
 - 用户影响: 直接影响使用 RDT 权重同步模式进行训练的用户。他们的训练脚本将无法启动。对于使用其他模式 (broadcast, p2p, disk-delta) 的用户无直接影响。
 - 系统影响: 简化了系统的权重更新路径和资源分配逻辑, 移除了一个复杂的、独立的传输协议栈。这降低了代码的即时复杂度, 但暂时移除了一个高性能 (基于 RDMA) 的选项。
 - 团队影响: 这是团队主导的一次大型重构的组成部分。它清除了旧有代码, 为采用更模块化、异步化的新架构扫清了障碍, 有利于后续开发。但需要团队确保 RDT 功能能在新架构上及时恢复。
- 风险标记: 核心路径变更, 缺少测试覆盖, 功能回退

关联脉络

- PR #1313 feat: RDT weight sync from megatron actor to sglang engine (#1313): 本 PR 明确声明是回滚 PR #1313 (RDT 功能的原始引入)。
- PR #1837 Tracking issue for refactoring and enhancements: 本 PR 是 #1837 重构跟踪 issue 下的一个子任务 (标记为 op2-11), 属于更大的重构工作流。

- PR #2729 feat: RDT multi-node support (#2729): commit message 提到回滚也包含了 PR #2729 的改动，这是对 RDT 多节点支持的后续增强。
- PR #1842 Split RolloutManager into InferenceController and RolloutExecutor: 从 #1837 issue 列表看，这是紧接着的、属于同一重构链 (op2-1) 的 PR，可能依赖或受益于此次 RDT 代码的清理。