PR #6067 完整报告

verl-project/verl

[BREAKING] [misc] refactor: deprecate workers, migrate to engines

合并时间：2026-04-20 21:00

原文链接：http://prhub.com.cn/verl-project/verl/pull/6067

执行摘要

- 一句话：废弃并删除整个 workers 模块，迁移至统一的模型引擎抽象。
- 推荐动作：该 PR 是理解项目架构演进的关键材料，值得技术管理者和核心工程师精读。重点关注：
  1. 设计决策：从分散的 worker 实现到统一引擎抽象的迁移策略，反映了项目在简化复杂性和提升模块化方面的思考。
  2. 集成模式：ray_trainer.py 中如何适配新引擎，特别是远程调用和错误处理模式，是分布式训练集成的典型案例。
  3. 破坏性变更管理：通过 @deprecated 装饰器和版本计划（v0.8.0）平滑过渡，展示了大型项目重构的最佳实践。

功能与动机

根据 PR body 描述，本次变更的核心动机是“减少维护负担，并促进模型引擎抽象和 verl 的 RL 库转型”。这表明项目正在进行架构演进，旨在用更统一的模型引擎抽象（如 TrainingWorker 和 ActorRolloutRefWorker）替代分散且复杂的传统 worker 实现，以简化代码库并提升可维护性。

实现拆解

1. 删除核心 worker 实现文件：移除了 verl/workers/ 目录下的所有核心源码文件，包括 fsdp_workers.py、megatron_workers.py 以及 Actor/Critic 的具体实现（如 dp_actor.py、megatron_actor.py、dp_critic.py、megatron_critic.py）。这些文件包含了 DataParallelPPOActor、MegatronPPOActor、DataParallelPPOCritic、MegatronPPOCritic 等关键类，它们负责 PPO 算法中策略和价值网络的前向计算、损失计算和参数更新。
2. 更新训练器逻辑以适配新引擎：修改了 verl/trainer/ppo/ray_trainer.py 等训练器文件，移除对 use_legacy_worker_impl 标志的依赖，并调整了 _compute_values、_compute_ref_log_prob 和 _update_actor 等方法，使其直接调用新的统一模型引擎（如 ActorRolloutRefWorker）进行远程计算。
3. 清理配置和依赖：移除了与 legacy worker 相关的配置项（如 use_legacy_worker_impl），并更新了导入路径，确保代码库不再引用已删除的模块。

4. 同步更新测试和文档：删除了与 workers 模块相关的测试文件（如 tests/workers/test_fsdp_attn_implementation.py）和文档（如 docs/workers/ 下的内容），确保测试套件和文档与新的架构保持一致。
5. 处理特殊模块的例外情况：根据 Issue 评论中的指示，保留了 verl/experimental/vla 模块，因为它计划迁移到独立仓库，避免了不必要的影响。

关键文件：

- verl/workers/fsdp_workers.py（模块 FSDP Worker；类别 source；类型 deletion；符号 create_device_mesh, get_sharding_strategy, get_vl_model_vision_tower, ActorRolloutRefWorker）：这是 FSDP 后端 worker 的核心实现文件，包含 ActorRolloutRefWorker 等关键类，负责模型构建、优化器初始化和 rollout 逻辑。其删除标志着 FSDP 路径的旧实现被彻底废弃。
- verl/workers/megatron_workers.py（模块 Megatron Worker；类别 source；类型 deletion；符号 set_random_seed, MegatronWorker, _init_hf_config_and_tf_config, ActorRolloutRefWorker）：这是 Megatron 后端 worker 的核心实现文件，包含 MegatronWorker 等关键类，负责 Megatron 特有的模型配置、优化器和训练循环。其删除标志着 Megatron 路径的旧实现被彻底废弃。
- verl/trainer/ppo/ray_trainer.py（模块 PPO 训练器；类别 source；类型 core-logic；符号 _compute_values, _compute_ref_log_prob, _update_actor）：这是 PPO 训练器的主入口文件，负责协调 worker 进行价值计算、参考策略概率计算和策略更新。本次变更移除了 legacy worker 实现路径，集成了新的统一模型引擎，是架构迁移的关键集成点。
- verl/workers/actor/dp_actor.py（模块 Actor 实现；类别 source；类型 deletion；符号 DataParallelPPOActor, init, _forward_micro_batch, _optimizer_step）：这是 DataParallel PPO Actor 的具体实现，包含 DataParallelPPOActor 类，负责 FSDP 数据并行下的策略网络前向计算和更新。其删除是 worker 模块清理的一部分。
- tests/workers/test_fsdp_attn_implementation.py（模块 Worker 测试；类别 test；类型 deletion；符号 TestFSDPAttnImplementation, test_attn_implementation_extraction_logic, test_attn_implementation_passed_to_autoconfig, test_attn_implementation_passed_to_model）：这是与 FSDP worker 相关的测试文件，其删除确保了测试套件与新的架构保持一致，避免了过时测试的干扰。

关键符号：DataParallelPPOActor, MegatronPPOActor, DataParallelPPOCritic, MegatronPPOCritic, _compute_values, _compute_ref_log_prob, _update_actor

关键源码片段

verl/trainer/ppo/ray_trainer.py

这是 PPO 训练器的主入口文件，负责协调 worker 进行价值计算、参考策略概率计算和策略更新。本次变更移除了 legacy worker 实现路径，集成了新的统一模型引擎，是架构迁移的关键集成点。

```
def _compute_ref_log_prob(self, batch: DataProto) -> DataProto:
    """计算参考策略的对数概率，适配新的统一模型引擎。"""
    # 根据配置决定使用哪个 worker group
```

```
if self.config.ref_in_actor:
    wg = self.actor_rollout_wg # 当参考策略与 Actor 融合时
else:
    # 关键修复：当使用新引擎且参考策略在 ActorRolloutRefWorker 中时，ref_policy_wg 可能为
    None
    wg = self.ref_policy_wg if self.ref_policy_wg is not None else self.actor_rollout_wg

# 远程调用 worker group 进行计算
output = wg.compute_ref_log_prob(batch)
# 关键修复：必须使用 .get() 收集远程结果，否则后续访问会出错
output = output.get() # 收集 RayDataProto 的实际内容

# 从输出中提取对数概率数据
log_prob = tu.get(output, "log_prob")
batch.batch["ref_log_prob"] = log_prob
return batch
```

评论区精华

review 讨论主要集中在 verl/trainer/ppo/ray_trainer.py 中与新的统一模型引擎集成的正确性问题上。

- 关键问题：*gemini-code-assist[bot]* 指出，在 _compute_ref_log_prob 方法中，当使用新引擎且参考策略与 ActorRolloutRefWorker 融合时，self.ref_policy_wg 可能为 None，导致逻辑错误。建议回退到 self.actor_rollout_wg。
- 远程调用处理：同一评论者多次强调，对 Ray worker 的远程调用结果（如 output）必须使用 .get() 收集后才能访问其内容，否则会在后续操作中引发错误。这涉及 _compute_values、_compute_ref_log_prob 和 _update_actor 方法。
- 结论：这些评论旨在修复集成漏洞，确保新引擎路径下的功能正确性。从提交历史看，作者通过多个提交逐步清理和修复，最终 PR 被合并，表明问题已得到解决。
  - 新引擎集成中的 worker group 回退逻辑 (correctness): 建议回退到 self.actor_rollout_wg，确保在新引擎路径下能正确选择 worker group。
  - 远程调用结果收集缺失 (correctness): 在 _compute_values、_compute_ref_log_prob 和 _update_actor 方法中添加了 .get() 调用，确保数据正确收集。

风险与影响

- 风险：
  1. 回归风险：删除大量核心 worker 代码（如 DataParallelPPOActor、MegatronPPOCritic）会直接影响所有依赖这些类的训练流程（特别是 FSDP 和 Megatron 后端）。如果新的统一引擎存在未覆盖的边缘情况或性能差异，可能导致训练失败或结果不一致。
  2. 集成风险：训练器（ray_trainer.py）中的逻辑调整，特别是远程调用结果未正确收集（如缺少 .get()）和 worker group 回退逻辑，可能引发运行时错误或数据丢失。

3. 兼容性风险：这是一个 BREAKING CHANGE，彻底移除了旧的 worker API，任何直接调用 verl.workers 模块的第三方代码或脚本将无法工作，需要迁移到新的引擎抽象。
4. 测试覆盖风险：虽然删除了旧的测试文件，但新引擎的测试覆盖是否充分尚不确定，可能隐藏未发现的缺陷。

- 影响：
  1. 对用户的影响：所有使用传统 worker 实现进行 PPO 训练的用户必须迁移到新的模型引擎抽象（如 TrainingWorker 和 ActorRolloutRefWorker）。这涉及更新配置、脚本和可能的自定义代码，迁移成本较高。
  2. 对系统的影响：简化了代码库结构，减少了维护负担，但短期内可能因新引擎的成熟度问题引入稳定性风险。统一抽象有望提升长期的可扩展性和一致性。
  3. 对团队的影响：标志着架构方向的重大转变，团队需要熟悉新的引擎 API 并更新相关文档和培训材料。 - 风险标记：核心路径变更，破坏性 API 变更，集成风险，测试覆盖调整

关联脉络

- PR #6074 [BREAKING] [env] refactor: deprecate verl/interactions: 类似的大规模废弃和删除操作，都涉及清理旧模块以简化代码库，反映了项目在架构演进中的一致性。
- PR #6053 [misc] fix: project name refactor - volcengine -> verl: 同为 misc 标签下的重构，涉及全局性的代码库调整，展示了项目在统一和标准化方面的持续努力。
- PR #6061 [veomni] feat: support Qwen3.5 SP and add GRPO trainer demo using VeOmniEngine: 涉及模型引擎（VeOmniEngine）的增强，与本 PR 推动的模型引擎抽象统一方向相关。