PR #1856 完整报告

THUDM/slime

refactor/ppo

合并时间：2026-04-24 23:00

原文链接：http://prhub.com.cn/THUDM/slime/pull/1856

执行摘要

- 一句话：PPO 训练架构重构，解耦 actor-critic 配置与通信
- 推荐动作：建议 PPO 训练用户深入阅读本 PR，理解新的 critic 配置和训练循环模型。值得关注的设计决策包括：移除 NCCL 同步改为 Ray 数据传递，以及强制 critic 与 actor 共享 GPU。这些设计简化了部署但损失了部分灵活性，需根据实际场景评估。

功能与动机

根据 PR 描述，此次重构为后续功能铺路，包括支持 actor 和 critic 不同并行配置、多 critic 等。当前架构中 actor 和 critic 强耦合，通过 NCCL 通信同步数据，限制了灵活性和扩展性。通过引入独立配置和异步 Ray 传参，实现解耦。

实现拆解

1. 引入 critic 独立配置机制：在 slime/utils/arguments.py 中新增 parse_critic_args 函数，通过 --critic-config-path 指定 YAML 文件，允许批评者拥有独立的 TP/PP 等并行配置。同时移除了旧的 --critic-load、--critic-save、--critic-lr 等独立 CLI 参数，保留 --num-critic-only-steps。新增 --custom-advantage-function-path 支持自定义优势函数。在 slime/backends/megatron_utils/arguments.py 中移除 critic_num_nodes/critic_num_gpus_per_node 相关逻辑，强制批评者与行动者共享相同的 GPU 资源。
2. 移除 actor-critic 间的 NCCL 进程组同步：在 slime/backends/megatron_utils/data.py 中删除 sync_actor_critic_data 函数，该函数通过 dist.broadcast 在 actor 和 critic 之间同步 values、log_probs 等张量。取而代之的是新增 tensors_to_cpu 和 tensors_to_gpu 工具函数，用于将张量移出 / 移入 GPU，以便通过 Ray 对象存储传递。在 slime/backends/megatron_utils/actor.py 的 train 方法中新增 external_data 参数，用于接收批评者传递的值。train_critic 方法返回 CPU 上的 values 字典，train_actor 则使用 external_data 中提供的 values 进行计算。
3. 重构训练循环与 Ray 传输层：在 train.py 中修改训练循环：批评者先执行 async_train 并获取 value_refs（Ray 对象引用），然后行动者训练时通过 external_data=value_refs 传递这些值。移除了 critic_train_only 条件分支，统一将 actor_trains_this_step 作为是否训练行动者的标志。在 train_async.py 中相应调整。在 slime/ray/actor_group.py 中修改 async_train 接口以支持 external_data，支持单个字典或列表形式。同时移除了 connect 方法，因为不再需要建立 NCCL 连接。

4. 调整资源分配与验证逻辑：在 slime/ray/placement_group.py 中，不再为批评者单独创建 placement group，而是与行动者共享同一组 GPU。create_placement_groups 返回的字典中 critic 键直接引用 actor。相应的，_compute_rollout_offset 等辅助函数移除了批评者相关的偏移计算。
5. 配套更新：更新了 examples/geo3k_vlm_multi_turn/run_geo3k_vlm_multi_turn_ppo_npu.py 以适配新的参数接口。同时在 slime/backends/megatron_utils/loss.py 中新增自定义优势函数支持，允许用户通过 --custom-advantage-function-path 提供自定义函数。

关键文件：

- slime/backends/megatron_utils/actor.py（模块 训练核心；类别 source；类型 core-logic；符号 train, train_critic, train_actor, connect_actor_critic）：核心训练逻辑重构，修改了 train、train_critic、train_actor 方法，新增 external_data 参数，移除了 connect_actor_critic 连接逻辑。
- slime/backends/megatron_utils/data.py（模块 数据传输；类别 source；类型 core-logic；符号 sync_actor_critic_data, tensors_to_cpu, tensors_to_gpu）：移除 sync_actor_critic_data 函数，新增 tensors_to_cpu 和 tensors_to_gpu 作为张量传输替代方案。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 dependency-wiring；符号 parse_critic_args）：新增 parse_critic_args 函数解析批评者 YAML 配置，移除旧 CLI 参数，新增 --custom-advantage-function-path。
- train.py（模块 训练入口；类别 source；类型 core-logic；符号 offload_train）：修改训练循环以支持新数据流，重构 offload_train 和 save 逻辑，传递 external_data。
- slime/ray/actor_group.py（模块 通信层；类别 source；类型 core-logic；符号 async_train, connect）：修改 async_train 以支持 external_data 参数，移除 connect 方法，适配新训练流程。
- slime/ray/placement_group.py（模块 资源调度；类别 source；类型 dependency-wiring）：修改资源分配，批评者与行动者共享 placement group，移除独立批评者 GPU 计算。
- slime/backends/megatron_utils/loss.py（模块 优势计算；类别 source；类型 core-logic）：添加自定义优势函数支持，在 compute_advantages_and_returns 中调用。
- train_async.py（模块 异步训练；类别 source；类型 core-logic）：适配训练循环变化，移除了 critic_train_only 条件。

关键符号：train, train_critic, train_actor, connect_actor_critic, sync_actor_critic_data, tensors_to_cpu, tensors_to_gpu, parse_critic_args, offload_train, async_train, connect, compute_advantages_and_returns

关键源码片段

train.py

修改训练循环以支持新数据流，重构 offload_train 和 save 逻辑，传递 external_data。

```
def offload_train(actor_trains_this_step):
    # 每个模型在 train() 之后自动卸载（当 offload_train 开启时），
    # 所以非 offload 情况下我们只需要清理内存即可。
```

```
if not args.offload_train:
    if not args.use_critic or actor_trains_this_step:
        actor_model.clear_memory()
    else:
        critic_model.clear_memory()
```

评论区精华

- 批评者值映射与并行限制：@Copilot 指出 critic_values_by_actor_worker 在非 last-PP-stage 工人键缺失时可能引发 KeyError。@zhuzilin 回应“如果只支持相同并行配置，可以简单传递数据，将复杂映射留给未来 PR”。当前仅支持相同配置，风险未完全解除。
- 资源调度风险：@Copilot 警告批评者与行动者共享 placement group 可能导致资源碎片 / 死锁。@zhuzilin 直接接受共享方案（result["critic"] = result["actor"]），认为当前可行。
- 代码健壮性：@Copilot 建议将 parse_critic_args 中的 assert 替换为 ValueError 以免禁用断言时静默失败。未见采纳。
- 传递方式设计：@zhuzilin 建议在训练循环中直接传递 value_refs（Ray ref）而非在控制进程中具体化。最终代码采用了此方式。
- 命名规范：@zhuzilin 要求遵循 --custom-xxx-function-path 命名约定，作者已调整。
- 未解疑虑：@zhuzilin 询问关闭 normalize advantage 的原因，无回复。
  - critic_values_by_actor_worker 潜在的 KeyError (correctness): 当前仅支持 actor 和 critic 相同并行配置，问题未完全解决但可接受。
  - 资源调度死锁风险 (performance): 接受共享，当前资源分配方式不改。
  - 训练循环中值传递方式 (design): 最终代码采用直接传递 value_refs 作为 external_data，符合建议。
  - 参数命名约定 (style): 作者调整了参数名（从 --custom-advantage-fn 改为 --custom-advantage-function-path）。
  - 为什么关闭 advantage 归一化 (question): 无回复，原因不明。

风险与影响

- 风险：
  1. 数值一致性风险：移除 NCCL broadcast 后，批评者值通过 Ray 对象存储 (CPU) 传递，再移回 GPU 时 tensors_to_gpu 转为 float32，可能引入精度差异，需验证 PPO 梯度更新不受影响。
  2. 资源调度死锁风险：批评者 placement group 引用行动者的 PG，在资源紧张时可能发生 gang scheduling 失败，建议增加重试或超时机制。
  3. 配置接口断裂：旧脚本若使用 --critic-load 等参数将失败，需要用户迁移到 YAML 配置文件。
  4. 自定义优势函数缺少验证：若用户提供的函数未正确设置 rollout_data['advantages']，后续代码将抛出 KeyError，定位困难。
  5. 代码路径覆盖：训练循环中的 actor_trains_this_step 逻辑改动可能影响 critic-only 或 PPO 配置特定阶段的行为，需要更全面的集成测试。

- 影响：
  - 用户影响：现有 PPO 训练用户需更新启动脚本以使用 --critic-config-path 替代旧的 critic 参数。自定义优势函数新增钩子提供扩展点。
  - 系统影响：训练数据流从 GPU NCCL 改为 CPU Ray 传输，可能增加端到端延迟但降低 GPU 间耦合，便于未来支持异构并行。
  - 团队影响：架构更加清晰，为后续多 critic、异步训练等特性打下基础。但重构范围广，需谨慎回归。
  - 影响程度：中等，主要涉及训练核心路径，不影响推理 /eval 流程。
  - 风险标记：训练循环重构，NCCL 同步移除，资源调度变化，配置接口变更，自定义优势函数未验证

关联脉络

- PR #1805 sync from internal: 涉及 Megatron model provider 和 SGLang rollout 的同步，与本 PR 的训练重构在同一功能线上。
- PR #1807 sync from internal: 涉及 Megatron model forward 参数构建，与本 PR 的训练核心改动有联系。