PR #1735 完整报告

radixark/miles

[PPO] Share Actor/Critic GPUs

合并时间：2026-07-31 08:33

原文链接：http://prhub.com.cn/radixark/miles/pull/1735

执行摘要

- 一句话：PPO 支持 Actor/Critic 共享 GPU 训练
- 推荐动作：值得精读。本 PR 展示了“driver 拥有 offload 计划、actor 拥有机制”的生命周期所有权拆分、基于 _asleep 的幂等 offload/onload（retry-safe）、用 Ray object reference 替代 NCCL 广播的数据交接方式，以及对 kl_coef（GAE 折扣）与 kl_loss_coef（KL 惩罚）语义差异的细致处理。建议重点关注 miles/backends/megatron_utils/actor.py 的 train/sleep/wake_up 与 miles/ray/placement_group.py 的布局计算，并跟踪后续“critic 独立参数、multi-critic、不同并行度”等 TODO 的演进。

功能与动机

PR body 明确目标是“Let PPO actor and critic time-share the same GPU placement through sequential offload/onload, eliminating dedicated critic GPU allocation”，即通过顺序 offload/onload 让 actor 与 critic 共享 GPU，从而省去专门给 critic 分配的卡。实现参考了 slime 仓库 #1856 的共享放置与 critic-value 交接设计，以及 #1882（offload 生命周期 bug）、#1888（checkpoint 保存时 resume/pause 缺失）、#1950（colocate 下 rollout GPU 计数错误）三个修复 PR。关联 Issue #1856 的 TODO 也明确指出“currently only support same parallel between actor and critic，maybe need more flexible”，说明当前方案先限定 actor/critic 并行拓扑一致，后续再做扩展。

实现拆解

1. 资源放置重构（miles/ray/placement_group.py）：新增纯函数 _get_placement_group_layout(args)，把原来散落在 create_placement_groups 里的分支逻辑（debug_train_only、debug_rollout_only、rollout_external、colocate）收敛为统一的 (num_gpus, rollout_offset) 计算；critic 的 placement group 直接复用 actor 的（result["critic"] = result["actor"]），colocate 下 PG 大小取 max(actor_num_gpus, rollout_num_gpus)，不再为 critic 追加 bundle；create_training_models 改为先串行 init actor、再 init critic，并用 copy.deepcopy(args) 生成 critic 专属参数（kl_coef=0、use_opd=False、disable_param_buffers_cpu_backup=False），同时删除 actor–critic 间的 connect() 调用。
2. value 传递改为 Ray object reference（miles/backends/megatron_utils/actor.py、miles/ray/actor_group.py）：删除 miles/backends/training_utils/data.py 中的 sync_actor_critic_data（原 NCCL broadcast 通道）及相关 process group 建立逻辑；

train_critic 在 pp_last_stage 返回 Box(ray.put([value.detach().cpu() for value inrollout_data["values"]])), train_actor 在同一 stage 用 ray.get(values_ref.inner) 取回并non_blocking 拷贝到当前设备；RayTrainGroup.train 增加 external_data 参数，支持按rank 分发 payload 列表并校验数量。

3. 生命周期显式化与幂等化（miles/backends/megatron_utils/actor.py）：sleep()/wake_up() 引入 _asleep 状态位变为幂等操作，避免重试或并发调用导致 torch_memory_saver 的 resume/pause 错乱；train() 与 save_model() 不再根据 use_critic/offload_train 隐式决定 offload，而是接收 driver 传入的 sleep_after_train/wake_up_before_save 选项，即“driver 拥有 offload 计划、actor 拥有机制”；save_model 内部不再自行 reload/destroy process group，update_weights 仅在 _asleep 时使用临时 process group。
4. 驱动主循环改造（train.py、train_async.py）：训练循环从“critic 与 actor 并发训练”改为串行——先 values = await critic_model.train(...)，再 actor_model.train(..., external_data=values)，每步训练后按需 offload；checkpoint 保存统一走 save_training_model(model)（先 onload、save_model、再 offload）；train_async.py 同步修复了旧的并发设计与 value 未传递问题；无 critic 时 critic_num_gpus_per_node/critic_num_nodes 不再默认继承 actor 尺寸。
5. 参数校验与测试配套（miles/utils/arguments.py、tests/…）：use_critic 时强制 Megatron backend、拒绝 bridge 模式 critic、保留显式 --no-offload-train（仅用于 offload 调试并告警）、拒绝 reward-level KL（--kl-coef != 0）以免 critic 训练目标错位；FSDP backend 接受新 save 生命周期选项；新增 test_shared_ppo_lifecycle.py、test_placement_group_shared_ppo.py、test_actor_group_shared_ppo.py 等测试，覆盖共享放置计数、critic value 按 rank 路由、生命周期幂等与 process group 行为。

关键文件：

- miles/backends/megatron_utils/actor.py（模块 训练执行；类别 source；类型 core-logic；符号 train, train_critic, train_actor, sleep）：核心执行与生命周期改造：train 按 role 分流并把 critic value 包装为 Ray object reference 传出；train_actor 在共享拓扑下对接 external_data；sleep/wake_up 幂等化；save_model/update_weights 的 process group 管理显式化。
- miles/ray/placement_group.py（模块 资源放置；类别 source；类型 core-logic；符号 _get_placement_group_layout, create_placement_groups, create_training_models）：资源放置重构入口：新增 _get_placement_group_layout 收敛布局计算，critic 直接复用 actor 的 placement group，colocate 下不再为 critic 单独加 bundle；create_training_models 改为串行 init 并构造 critic 专属参数。
- miles/ray/actor_group.py（模块 训练组；类别 source；类型 core-logic；符号 train, connect）：RayTrainGroup.train 支持 external_data 按 rank 分发，是 critic value 从驱动到各 actor worker 的通道；同时移除已废弃的 connect()（actor–critic 通信组建立）。
- train.py（模块 训练主循环；类别 source；类型 core-logic；符号 save_training_model）：驱动主循环：critic 训练先行并把 values 传给 actor，训练后逐个 offload；save_training_model 统一处理 checkpoint 保存时的 onload/offload 序列。

- miles/backends/training_utils/data.py（模块 数据同步；类别 source；类型 core-logic；符号 sync_actor_critic_data）：删除 sync_actor_critic_data，宣告 actor–critic NCCL 同步通道的退役，是架构变更的直接证据。
- tests/fast/backends/megatron_utils/test_shared_ppo_lifecycle.py（模块 生命周期测试；类别 test；类型 test-coverage；符号 actor_module, _worker, test_critic_train_wakes_and_leaves_offload_to_driver, test_actor_receives_critic_payload_and_leaves_offload_to_driver）：新增生命周期专项测试：覆盖 critic/actor train 的 wake 行为与 offload 归属、save_model 不管理生命周期、update_weights 仅在 asleep 时用临时 process group、sleep/wake_up 幂等性。

关键符号：train, train_critic, train_actor, sleep, wake_up, save_model, update_weights, connect_actor_critic, connect, sync_actor_critic_data, _get_placement_group_layout, create_placement_groups, create_training_models, save_training_model, start_mooncake_master

关键源码片段

miles/backends/megatron_utils/actor.py

核心执行与生命周期改造：train 按 role 分流并把 critic value 包装为 Ray object reference 传出；train_actor 在共享拓扑下对接 external_data；sleep/wake_up 幂等化；save_model/update_weights 的 process group 管理显式化。

```
# miles/backends/megatron_utils/actor.py
# 共享 GPU 后，critic 与 actor 分时占用同一批物理卡，sleep/wake_up 成为高频切换点。
# 引入 _asleep 状态位后两个方法都变成幂等操作，避免重试或并发调用导致 resume/pause 错乱。

@with_logs
@timer
def sleep(self) -> None:
    assert self.args.offload_train
    if self._asleep:
        logger.info("sleep() called while already offloaded; skipping")
        return

    clear_memory(clear_host_memory=True)
    print_memory("before offload model")
    should_log_cpu_memory = is_first_replica_megatron_main_rank() and hasattr(self, "_last_
    rollout_id")

    destroy_process_groups()

    tag = "default" if is_lora_enabled(self.args) else None
    torch_memory_saver.pause(tag=tag)

    self._asleep = True
    print_memory("after offload model")

    if should_log_cpu_memory:
```

```
        log_cpu_memory(self._last_rollout_id, self.args, "after_offload_train")


    @with_logs
    @timer
    def wake_up(self) -> None:
        assert self.args.offload_train
        if not self._asleep:
            # 模型常驻（例如重试在 wake 与 sleep 之间失败）时无需 resume，但
            # process group 可能已被销毁，因此必须恢复 group 以保证后续通信可用。
            logger.info("wake_up() called while already resident; ensuring process groups only")
            reload_process_groups()
            return
        print_memory("before wake_up model")

        tag = "default" if is_lora_enabled(self.args) else None
        torch_memory_saver.resume(tag=tag)

        clear_memory()
        reload_process_groups()
        self._asleep = False
        print_memory("after wake_up model")
```

miles/ray/placement_group.py

资源放置重构入口：新增 _get_placement_group_layout 收敛布局计算，critic 直接复用 actor 的 placement group，colocate 下不再为 critic 单独加 bundle；create_training_models 改为串行 init 并构造 critic 专属参数。

```
# miles/ray/placement_group.py
# 布局计算收敛为纯函数：critic 不再单独占卡，colocate 下 PG 大小取
# actor 与 rollout 的较大者，rollout 与训练共享同一批物理 GPU。

def _get_placement_group_layout(args) -> tuple[int, int]:
    """返回 (num_gpus, rollout_offset)，由 create_placement_groups 统一消费。"""
    actor_num_gpus = args.actor_num_nodes * args.actor_num_gpus_per_node

    if args.debug_train_only:
        return actor_num_gpus, 0
    if args.rollout_external:
        # 外部 rollout：只在需要时给本地训练保留 actor 大小的卡
        if args.debug_rollout_only:
            return 0, 0
        return actor_num_gpus, actor_num_gpus
    if args.debug_rollout_only:
        return args.rollout_num_gpus, 0
    if args.colocate:
        # actor/critic/rollout 分时共享，取两者较大值即可
        return max(actor_num_gpus, args.rollout_num_gpus), 0
    return actor_num_gpus + args.rollout_num_gpus, actor_num_gpus
```

```
async def create_training_models(args, pgs, rollout_manager):
    actor_model = allocate_train_group(
        args=args,
        num_nodes=args.actor_num_nodes,
        num_gpus_per_node=args.actor_num_gpus_per_node,
        pg=pgs["actor"],
        role="actor",
        with_ref=args.kl_coef != 0 or args.use_kl_loss,
        rollout_manager=rollout_manager,
        with_opd_teacher=args.use_opd and args.opd_type == "megatron",
    )
    actor_start_rollout_ids = await actor_model.init()

    if args.use_critic:
        # critic 继承 actor 参数，但关闭 reward 级 KL 与 OPD，
        # 因为共享 GPU 后不再有 NCCL 通道把相关张量从 actor 送到 critic。
        critic_args = copy.deepcopy(args)
        critic_args.kl_coef = 0
        critic_args.use_opd = False
        critic_args.disable_param_buffers_cpu_backup = False
        critic_model = allocate_train_group(
            args=critic_args,
            num_nodes=args.critic_num_nodes,
            num_gpus_per_node=args.critic_num_gpus_per_node,
            pg=pgs["critic"], # 与 actor 共享同一 placement group
            role="critic",
            with_ref=False,
            rollout_manager=None,
        )
        critic_start_rollout_ids = await critic_model.init()
    else:
        critic_model = None

    start_rollout_ids = critic_start_rollout_ids if args.use_critic else actor_start_rollout_ids
    assert len(set(start_rollout_ids)) == 1
    if args.start_rollout_id is None:
        args.start_rollout_id = start_rollout_ids[0]

    await actor_model.set_rollout_manager()
    if args.rollout_global_dataset:
        await rollout_manager.load.remote(args.start_rollout_id - 1)

    return actor_model, critic_model
```

评论区精华

生命周期归属：guapisolo 在 actor.py 的 review 中建议 train 增加 sleep: bool = False 参数、save_model 增加 wake_up 参数，让 driver 显式控制 offload 时机（“control this flag in train.py / train_async.py when critic is used. This can make sure code logic is

self-included”）；后续 commit drive shared actor/critic lifecycle via driver options 落实了该设计。

assert vs ValueError：gemini-code-assist[bot] 指出 -O 会全局禁用 assert，建议配置校验改用 ValueError（涉及 arguments.py、placement_group.py 的 start rollout id 校验）；Shi-Dong 评论“Might be worth fixing”。部分校验后续保留为 assert（例如 value 到达的硬不变量），配置类校验则转向 fail loud。

kl_coef 语义陷阱：yueming-yuan 在 placement_group.py 问是否应加 kl_coef == 0 断言；guapisolo 确认“kl_coef 与 kl_loss_coef 是两回事，前者应用于 ppo gae，后者用于 kl penalty”，并据此提交 reject reward-level KL under shared actor/critic：colocate 重构删掉了把 actor log_probs/ref_log_probs 送到 critic 的 NCCL 通道，critic 无法再折叠 --kl-coef 奖励惩罚，因此直接 fail loud，而不是静默训练错位目标。

values_ref 硬断言：yueming-yuan 问“会不会出现 external_data 非空且 pp_last_stage 但 values_ref 为 None 的情况，若是应断言并移除 if 分支”；guapisolo 随后提交 assert critic values reach the pp-last-stage actor rank，把“critic 与 actor 拓扑相同则 paired rank 必有 values”作为不变量。

ft v2 兼容：guapisolo 在 actor_group.py 的 train 签名变更处自评“This might make ft v2 crash. I will push a fix.”，后续通过让 train() 兼容无 external_data 的广播路径解决。

E2E 配置：guapisolo 建议共享后 tests/e2e/megatron/test_qwen3_4B_ppo.py 可把 PP size 提高到 2，xiuhu17 采纳。

- train()/save_model() 增加生命周期参数，由 driver 控制 offload (design)：采纳；后续 commit “drive shared actor/critic lifecycle via driver options”实现 driver 传选项、actor 只提供机制的拆分。
- 配置校验应使用 ValueError 而非 assert (correctness)：部分采纳：配置类校验改为 fail loud；训练内不变量（如 values 必须到达 pp_last_stage）仍保留 assert。
- 共享模式下 kl_coef 被强制 0 导致 reward-level KL 目标错位 (correctness)：采纳并 fail loud：拒绝 --kl-coef != 0 的 PPO 配置，保留 --use-kl-loss 作为 KL 正则。
- values_ref 是否可能为 None，应断言而非静默跳过 (correctness)：采纳；guapisolo 提交“assert critic values reach the pp-last-stage actor rank”，把配对 rank 必有 values 作为不变量。
- actor_group.train 签名变化可能使 ft v2 crash (correctness)：已通过保留无 external_data 的广播路径并兼容旧调用修复。
- E2E 共享后可提高 PP size (testing)：已采纳并修改 E2E 配置。

风险与影响

- 风险：生命周期状态机风险：共享 GPU 的正确性完全依赖 offload/onload 的顺序与幂等，_asleep 状态若与真实显存状态不同步，会出现 torch_memory_saver.resume 对已 resident 模型误调用或对已 paused 模型继续训练；slime #1888 的 checkpoint crash（save 时缺 resume 触发 CUDA error: invalid argument）即属此类，save_model 路径不再自行管理 process group 后，driver 的 onload/offload 顺序必须严格保证。 数据契约风

险：critic value 通过 Ray object reference 传递依赖"actor/critic 共享相同并行拓扑"的假设，train_actor 在 pp_last_stage 用 assert 强制 values 存在；一旦未来支持不同并行度或非 pp_last_stage 的 value 来源，该契约会立即失效。 性能与内存：value 张量 detach().cpu() 后进 Ray object store，大 batch、长序列下会有 CPU 内存与序列化开销；actor 端 non_blocking 拷贝叠加在训练关键路径上。另 --no-offload-train 被保留后，双模型常驻共享卡可能直接 OOM（仅建议 offload 调试用）。 行为兼容：--kl-coef != 0 与 PPO 现在直接报错；bridge 模式拒绝 critic；critic_num_gpus_per_node/critic_num_nodes 在无 critic 时不再默认设置；FSDP 与实验性 ft trainer 需适配新生命周期选项，actor_group.train 签名变化曾引发 ft v2 crash 预警。 CI 环境：issue 评论中确认 CI 镜像（SGLang 0.5.15）与滚动源码（0.5.16）依赖漂移、Docker 内多 cuDNN 版本导致 GPT-OSS 的 TE fused attention 失败，属于环境问题，但阻塞了本 PR 的验证周期。

- 影响：资源影响：PPO 训练不再独立为 critic 分配 GPU，actor/critic/rollout 分时共享同一批物理卡，卡资源需求大幅下降；PR 验证中 4×H200 可完成 3 个 rollouts 的 actor+critic 全量训练，Qwen3-4B-Thinking 30 步 PPO 的 reward 曲线与旧实现一致。系统影响：训练主循环由 critic/actor 并发改为严格串行（critic 先、actor 后），actor–critic 之间的 NCCL 通信组被移除，分布式耦合降低，但单步训练时延可能因串行与 offload 而增加。 团队影响：train.py、train_async.py、arguments.py、placement_group.py 的行为与参数语义均有变化，依赖 shared-PPO 的脚本需检查 --kl-coef、--no-offload-train、bridge 模式等配置；FSDP、ft v2 等后端需适配统一的生命周期选项；新增的生命周期与 placement 测试为后续回归提供保障。
- 风险标记：核心路径变更，生命周期状态机，数据契约变更，跨后端兼容，CI 环境依赖

关联脉络

- PR #2014 fix: quantize non-interleaved DSA indexer wk: 本 PR 的 PPO E2E 在 dpsk v32 fp8 weight update 阶段出现问题，issue 评论中 xiuhu17 确认由 #2014 修复，二者同属 PPO 训练链路的质量修复。
- PR #2028 session: collect speculative-decoding counters: 同为训练 /rollout 会话观测配套改动，属于 PPO 训练链路的 session 数据采集演进，与本 PR 的 train 主循环改造有间接关联。