PR #2180 完整报告

THUDM/slime

Add --release-train

合并时间：2026-07-06 14:52

原文链接：http://prhub.com.cn/THUDM/slime/pull/2180

执行摘要

- 一句话：新增 --release-train 支持训练后释放 actor 组并重建
- 推荐动作：建议 值得精读，特别是 slime/ray/actor_group.py 中 release() 和 create() 的实现思路，以及 train.py 中训练循环的重构。设计上将资源生命周期与训练步骤解耦，为资源弹性调度提供了参考。

功能与动机

PR 标题和代码注释未提供明确动机，但从变更内容推断，为了在训练间隙释放 GPU 资源给其他任务使用（如 Rollout），并支持从 Checkpoint 恢复训练，从而提升集群资源利用率和训练灵活性。

实现拆解

1. 新增命令行参数 --release-train（slime/utils/arguments.py）：添加布尔标志，控制是否启用训练释放机制。
2. 重构 RayTrainGroup（slime/ray/actor_group.py）：将 Actor 的 GPU 分配和初始化从 __init__ 推迟到新的 create() 方法；添加 release() 方法用于销毁 Actor 并释放 Placement Group；修改 save_model() 和 update_weights()，在启用 release-train 时在保存后设置加载路径并调用 release() 和 _reload_rollout_weights_from_disk() 重新加载 Rollout 权重。
3. 提取 create_actor_model 函数（slime/ray/placement_group.py）：从 create_training_models 中独立出 Actor 模型创建逻辑，以便在训练循环中重新创建；allocate_train_group 增加 with_ref / with_opd_teacher 参数传递。
4. 修改训练入口（train.py / train_async.py）：在每次训练迭代开始前，若启用 release-train 则调用 actor_model.create() 重建 Actor；保存模型时强制同步并触发释放逻辑。
5. SGLang HiCache 内存释放补丁（docker/patch/latest/sglang-release_hicache.patch）：为 SGLang 调度器添加 release_hicache_memory / resume_hicache_memory 方法，在权重更新释放 KV Cache 时同步释放 HiCache 宿主内存。
6. 端到端测试（tests/test_release_train.py）：模拟 Colocate 模式下两个 Rollout 步骤，验证 Actor 释放后能从 Checkpoint 正确恢复。

关键文件：

- slime/ray/actor_group.py（模块 集群调度；类别 source；类型 dependency-wiring；符号 async_init, release, create, _release_train_enabled）：核心重构文件：实现 actor 组的延迟创建和释放，以及 release-train 的核心逻辑。
- slime/ray/placement_group.py（模块 集群调度；类别 source；类型 core-logic；符号 allocate_train_group, create_training_models, create_actor_model）：提取 create_actor_model 函数，重构训练模型创建流程，支持重新创建 actor。
- train.py（模块 训练入口；类别 source；类型 core-logic；符号 save）：训练入口核心修改：在循环中支持重新创建 actor 和强制同步保存。
- tests/test_release_train.py（模块 集成测试；类别 test；类型 test-coverage；符号 prepare, execute）：新增端到端冒烟测试，验证 release-train 流程的正确性。
- docker/patch/latest/sglang-release_hicache.patch（模块 SGLang 补丁；类别 infra；类型 test-coverage；符号 init_weight_updater, release_hicache_memory, resume_hicache_memory, write_backup）：新增的 SGLang 补丁，在权重更新时释放 HiCache 主机内存，是 release-train 的配套组件。
- slime/utils/arguments.py（模块 参数配置；类别 source；类型 configuration）：添加 --release-train 命令行参数入口。

关键符号：save_model, update_weights, release, create, _release_train_enabled, _full_disk_weight_update_enabled, _reload_rollout_weights_from_disk, allocate_train_group, create_actor_model, prepare, execute

关键源码片段

slime/ray/actor_group.py

核心重构文件：实现 actor 组的延迟创建和释放，以及 release-train 的核心逻辑。

```
  def update_weights(self):
      """Broadcast weights from rank 0 to all other ranks."""
      # 如果不是全量磁盘更新，则沿用原有路径
      if not self._full_disk_weight_update_enabled():
          return ray.get([actor.update_weights.remote() for actor in self._actor_handlers])

      # 全量磁盘更新时，先更新权重，然后判断是否需要释放 actor 组
      weight_version = self._disk_weight_version + 1
      disk_weight_dir = Path(self.args.update_weight_disk_dir) / f"weight_v{weight_version:06d}"
      ray.get([actor.update_weights.remote() for actor in self._actor_handlers])
      self._disk_weight_version = weight_version
      if self._release_train_enabled():
          # 释放 actor 组，释放 GPU 资源
          self.release()
          # 从磁盘重新加载 rollout 权重
          self._reload_rollout_weights_from_disk(disk_weight_dir, str(weight_version))


  def save_model(self, rollout_id, force_sync=False):
      """Save actor model"""
```

```
    ret = ray.get([actor.save_model.remote(rollout_id, force_sync=force_sync) for actor in self._
    actor_handlers])
    if self._release_train_enabled():
        # 如果启用了 release-train，则在保存后设置加载路径为当前保存路径
        self.args.load = self.args.save
        self.args.ckpt_step = None
        self.args.finetune = False
        self.args.no_load_optim = self.args.no_save_optim
        self.args.no_load_rng = False
    return ret
```

slime/ray/placement_group.py

提取 create_actor_model 函数，重构训练模型创建流程，支持重新创建 actor。

```
def create_actor_model(args, pgs, rollout_manager, actor_cls=None):
    """独立创建 actor 模型并初始化，返回模型实例和初始 rollout id"""
    actor_args = args
    if args.megatron_config_path is not None:
        from slime.utils.arguments import parse_megatron_role_args
        actor_args = parse_megatron_role_args(args, args.megatron_config_path, role="actor")

    actor_model_kwargs = {}
    if actor_cls is not None:
        actor_model_kwargs["actor_cls"] = actor_cls

    actor_model = allocate_train_group(
        args=actor_args,
        num_nodes=args.actor_num_nodes,
        num_gpus_per_node=args.actor_num_gpus_per_node,
        pg=pgs["actor"],
        with_ref=actor_args.kl_coef != 0 or actor_args.use_kl_loss,
        with_opd_teacher=actor_args.use_opd and actor_args.opd_type == "megatron",
        **actor_model_kwargs,
    )
    # 调用新的 create 方法（代替原来的 async_init）
    actor_start_rollout_ids = actor_model.create(rollout_manager=rollout_manager)
    return actor_model, actor_start_rollout_ids
```

train.py

训练入口核心修改：在循环中支持重新创建 actor 和强制同步保存。

```
# 训练循环
for rollout_id in range(args.start_rollout_id, args.num_rollout):
    # ... 省略 rollout 采样部分
    # 如果启用了 release-train，则重新创建 actor（从 checkpoint 恢复）
    if release_train:
        actor_model.create()

    actor_trains = (not args.use_critic) or rollout_id >= args.num_critic_only_steps
```

```
    # ... 训练调用省略

    # 保存模型：release-train 模式下每次迭代都保存并触发释放
    if release_train or should_run_periodic_action(...):
        force_sync = release_train or rollout_id == args.num_rollout - 1
        if actor_trains:
            actor_model.save_model(rollout_id, force_sync=force_sync)
        if args.use_critic:
            critic_model.save_model(rollout_id, force_sync=force_sync)
        if args.rollout_global_dataset:
            ray.get(rollout_manager.save.remote(rollout_id))
    # ... 后续 offload 和权重更新
```

评论区精华

本 PR 由作者自行合并，未发现公开 review 讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 训练中断风险：若磁盘权重更新或 Checkpoint 保存失败，Actor 组可能已释放，导致训练无法恢复。需要保证 save 和 update_weights 的原子性。
  2. 兼容性风险：--release-train 依赖于 Colocate 模式和磁盘权重更新，与其他配置（如 NIXL 传输、非全量更新）的交互未充分测试。
  3. 性能开销：每次迭代的释放与重建引入额外延迟，在短迭代中可能得不偿失。
  4. Checkpoint 一致性：重建 Actor 时若 load 路径指向不完整 Checkpoint，可能导致状态丢失（如 optimizer 状态、RNG 状态）。
  5. SGLang 补丁冲突：新增的 HiCache 补丁与现有 sglang.patch 可能存在冲突，需验证打补丁顺序。 - 影响：用户视角：新增 --release-train 可选参数，启动训练时加入即可启用。启用后训练流程会在每次保存后释放 GPU 资源，适合共享集群场景。系统视角：核心训练流程（train.py）和 Actor 管理（actor_group.py）均受影响，变更覆盖 17 个文件，新增 912 行、删除 183 行。需要配套升级 SGLang 补丁。团队视角：为未来支持训练与 Rollout 更灵活的 GPU 复用奠定了基础，但也增加了系统复杂度和维护成本。
- 风险标记：核心路径变更，新增配置项，资源释放风险，SGLang 补丁新增

关联脉络

- PR #2089 Disk-level delta weight sync: release-train 依赖于磁盘全量权重更新机制，该 PR 实现了磁盘级权重同步的基础设施。
- PR #2134 fix: handle empty colocated weight buckets: 修复了 colocated 权重桶为空的 Bug，与 release-train 共用的 Actor 组管理逻辑有交叉。