PR #5967 完整报告

verl-project/verl

[fully_async] fix: Add Mindspeed Patch for Async Training on Ascend NPUs

合并时间：2026-04-20 13:54

原文链接：http://prhub.com.cn/verl-project/verl/pull/5967

执行摘要

- 一句话：修复完全异步训练在 Ascend NPU 上因 MindSpeed 补丁未应用导致的 torch_npu 错误。
- 推荐动作：该 PR 值得精读，特别是对于在 Ascend NPU 上使用完全异步训练的开发者。关注点包括：
  - 设计决策：将策略处理器改为属性方法以控制导入顺序，这是一种常见的延迟初始化模式，适用于依赖外部补丁的场景。
  - 代码组织：注意 review 中提到的补丁位置重构建议，这可能影响未来代码维护。
  - 关联风险：了解 init_model 中未解决的配置访问问题，避免在非 Megatron 策略下误用。

功能与动机

根据 PR body 和 review 评论，动机是解决在 Ascend NPU 上进行完全异步训练时出现的 torch_npu 错误。问题根源是 DetachActorWorker 在初始化时立即调用 _get_strategy_handlers()，导致 megatron 模块过早导入，使得 MindSpeed 的猴子补丁无法生效。这影响了完全异步训练路径在 NPU 上的正常运行。

实现拆解

1. 修改 DetachActorWorker 的策略处理器加载逻辑：
   - 文件：verl/experimental/separation/engine_workers.py
   - 关键符号：copy_handler、restore_handler
   - 变更：移除 __init__ 中直接调用 _get_strategy_handlers() 的代码，将 copy_handler 和 restore_handler 改为 @property 装饰的属性方法，实现延迟加载。
   - 原因：避免在初始化时过早导入 verl.utils.megatron_utils 模块，从而确保 MindSpeed 补丁能在 megatron 模块导入前正确应用。
   - 影响：修复了完全异步训练在 NPU 上的兼容性问题，使 save_model_to_cpu 和 restore_model_from_cpu 方法能正常调用处理器。
2. 新增完全异步训练脚本示例：
   - 文件：verl/experimental/fully_async_policy/shell/geo3k_qwen3vl_30b_megatron_6_2_npu_async.sh
   - 关键符号：无

- 变更：新增一个 shell 脚本，配置了使用 Megatron 策略在 Ascend NPU 上训练 Qwen3-VL-30B 模型的完全异步训练参数。
- 原因：提供修复后的使用示例，展示如何在实际场景中应用此修复。
- 影响：帮助用户快速上手在 NPU 上进行完全异步训练。

3. 其他配套调整：
   - 在 verl/experimental/separation/engine_workers.py 中添加了 from typing import Callable 导入，以支持类型注解。
   - 无测试或配置文件的直接改动，但修复确保了现有测试和配置在 NPU 环境下的正确性。

关键文件：

- verl/experimental/separation/engine_workers.py（模块 分离引擎；类别 source；类型 core-logic；符号 copy_handler, restore_handler）：核心修复文件，修改了 DetachActorWorker 的策略处理器加载逻辑，解决了 MindSpeed 补丁导入顺序问题。
- verl/experimental/fully_async_policy/shell/geo3k_qwen3vl_30b_megatron_6_2_npu_async.sh（模块 异步策略；类别 other；类型 configuration）：新增的示例脚本，展示了修复后在 Ascend NPU 上使用 Megatron 策略进行完全异步训练的具体配置。

关键符号：copy_handler, restore_handler, _get_strategy_handlers, save_model_to_cpu, restore_model_from_cpu

关键源码片段

verl/experimental/separation/engine_workers.py

核心修复文件，修改了 DetachActorWorker 的策略处理器加载逻辑，解决了 MindSpeed 补丁导入顺序问题。

```
class DetachActorWorker(ActorRolloutRefWorker):
    # ... 其他代码 ...

    def __init__(self, config: DictConfig, role: str):
        ActorRolloutRefWorker.__init__(self, config, role)
        self._strategy_handlers = None # 移除直接调用 _get_strategy_handlers()，避免过早导入

    def _get_strategy_handlers(self):
        # 延迟加载策略处理器，支持 FSDP、FSDP2 和 Megatron
        if self._strategy_handlers is not None:
            return self._strategy_handlers
        strategy = self.config.actor.strategy
        if strategy in ["fsdp", "fsdp2"]:
            from verl.utils.fsdp_utils import fsdp2_sharded_save_to_cpu, fsdp2_sharded_load_from_
            cpu
            self._strategy_handlers = (fsdp2_sharded_save_to_cpu, fsdp2_sharded_load_from_cpu)
        elif strategy == "megatron":
            from verl.utils.megatron_utils import copy_megatron_model_to_cpu, restore_megatron_
            model_from_cpu
            self._strategy_handlers = (copy_megatron_model_to_cpu, restore_megatron_model_
```

```
            from_cpu)
        else:
            raise NotImplementedError(f"Unsupported strategy: {strategy}")
        return self._strategy_handlers

    @property
    def copy_handler(self) -> Callable:
        """
        获取策略对应的复制处理器。
        通过属性延迟加载，确保在需要时才导入相关模块，避免干扰MindSpeed补丁。
        """
        return self._get_strategy_handlers()[0]

    @property
    def restore_handler(self) -> Callable:
        """获取策略对应的恢复处理器。"""
        return self._get_strategy_handlers()[1]

    @register(dispatch_mode=Dispatch.ONE_TO_ALL)
    def save_model_to_cpu(self, n):
        if not hasattr(self, "cpu_saved_models"):
            self.cpu_saved_models = {}
        self.cpu_saved_models[n] = self.copy_handler(self.actor.engine.module) #
        使用属性方法调用处理器
```

评论区精华

review 讨论主要集中在 MindSpeed 补丁的应用时机和安全性上：

- gemini-code-assist[bot] 指出潜在运行时崩溃风险：在 verl/workers/engine_workers.py 的 init_model 方法中，直接访问 self.config.actor.megatron 可能在不使用 Megatron 策略时导致 AttributeError 或 KeyError，建议添加防护条件。
- HwCARI 解释了问题根源：DetachActorWorker.__init__() 过早调用 _get_strategy_handlers()，导致 megatron 模块在 MindSpeed 补丁前导入，使补丁失效。
- wuxibin89 建议重构补丁位置：提议将 repatch 函数移到 verl/workers/engine/mindspeed/transformer_impl.py 中，以更好地组织代码。
- 结论：PR 通过延迟加载策略处理器解决了导入顺序问题，但 review 中提到的 init_model 中的安全风险和补丁重构建议未在本 PR 中解决，可能留待后续处理。
  - MindSpeed 补丁导入顺序问题 (correctness): 通过将 copy_handler 和 restore_handler 改为属性方法实现延迟加载，解决了导入顺序问题。
  - init_model 中配置访问安全性 (correctness): 未在本 PR 中解决，建议添加防护条件或后续重构。
  - 补丁代码组织 (design): 未在本 PR 中实施，可能作为未来重构任务。

风险与影响

- 风险：技术风险较低，主要涉及：
  - 回归风险：修改 copy_handler 和 restore_handler 为属性方法，可能影响其他依赖这些属性的代码，但变更保持了接口一致性，风险可控。
  - 性能风险：延迟加载可能导致首次调用时轻微延迟，但避免了不必要的早期导入，整体影响可忽略。
  - 兼容性风险：修复针对 Ascend NPU 和 Megatron 策略，可能不适用于其他硬件或策略，但未破坏现有功能。
  - 安全风险：无直接安全影响。
  - 未解决风险：review 中提到的 init_model 方法中的配置访问安全问题仍未处理，可能在非 Megatron 策略下引发崩溃。
- 影响：影响范围有限但关键：
  - 对用户的影响：修复后，用户可以在 Ascend NPU 上正常使用完全异步训练功能，避免 torch_npu 错误，提升 NPU 训练的可用性。
  - 对系统的影响：仅影响使用 Megatron 策略的完全异步训练路径，对 FSDP 等其他策略无影响，系统其他部分保持稳定。
  - 对团队的影响：提供了 NPU 异步训练的示例脚本，降低了团队在 NPU 环境下的配置和调试成本。
  - 影响程度：中等，解决了特定硬件和训练模式下的阻塞问题，但未引入新功能或架构变更。
  - 风险标记：导入顺序敏感，配置访问未防护

关联脉络

- PR #6052 [fully_async] fix: avoid blocking ray.get inside async actor methods: 同属完全异步训练模块的修复，涉及异步训练中的阻塞问题，与本 PR 的 NPU 兼容性修复互补。
- PR #6046 [fully_async] fix: preserve per-iteration routed_experts on partial rollout resume: 同属完全异步训练模块的修复，关注 rollout 恢复逻辑，与本 PR 共同提升完全异步训练的稳定性。
- PR #6029 [fully_async] fix: replace routed_experts on partial rollout resume i…: 同属完全异步训练模块的修复，涉及 MoE 路由专家问题，显示团队在持续优化完全异步训练路径。
- PR #6012 [fully_async] fix: add fully async grpo qwen3-235b npu script in main branch: 类似地提供了 NPU 上的完全异步训练脚本，与本 PR 新增的脚本示例在目的和硬件环境上相关。