PR #6072 完整报告

verl-project/verl

[veomni] feat: enable VeOmni engine for on-policy distillation

合并时间：2026-04-21 11:16

原文链接：http://prhub.com.cn/verl-project/verl/pull/6072

执行摘要

- 一句话：为在线策略蒸馏（OPD）启用 VeOmni 引擎支持，扩展训练后端兼容性。
- 推荐动作：该 PR 值得精读，特别是对于关注训练后端扩展和蒸馏集成的工程师。值得关注的设计决策包括： 1) VeOmni 复用 FSDP2 路径的兼容性设计（基于继承关系）； 2) 构造函数签名对齐的修复方式，体现了对父类接口的尊重； 3) 通过注释明确记录已知注意事项（如设备不匹配），提升了代码可维护性。建议结合示例脚本理解完整工作流。

功能与动机

根据 PR 描述，目标是“为蒸馏和分离工作器组件添加 VeOmni 策略支持，以便 VeOmniEngine 可以在在线策略蒸馏（OPD）场景中用作训练后端”。这旨在扩展 Verl 框架的训练后端选项，利用 VeOmni 引擎的特性（如继承自 FSDP2 的 DTensor 兼容性）来支持 OPD 工作流。

实现拆解

1. 扩展分离工作器的策略支持：修改 verl/experimental/separation/engine_workers.py 中的 DetachActorWorker 类。在 __init__ 方法中新增 distillation_config 参数并传递 **kwargs 给父类，以保持与父类 ActorRolloutRefWorker 的签名兼容性。在 _get_strategy_handlers 方法中，将 "veomni" 添加到策略列表中，使其复用 FSDP2 的分片保存 / 加载工具（fsdp2_sharded_save_to_cpu 和 fsdp2_sharded_load_from_cpu），因为 VeOmni 内部使用 FSDP2 进行数据并行。同时更新类文档字符串和 save_model_to_cpu/restore_model_from_cpu 方法的注释，说明 VeOmni 的兼容性注意事项（如 param_offload=True 时的设备不匹配风险）。
2. 适配蒸馏损失计算：修改 verl/trainer/distillation/losses.py 中的 compute_topk_loss 函数。在 match config.strategy: 语句中，将 case "fsdp": 改为 case "fsdp" | "veomni":，使 VeOmni 策略复用 FSDP 的损失计算实现（fsdp_losses.compute_forward_kl_topk），因为 VeOmni 内部使用 FSDP2，其损失计算逻辑与 FSDP 相同。
3. 新增示例脚本：创建 examples/on_policy_distillation_trainer/run_qwen_gsm8k_veomni.sh 脚本。该脚本提供了使用 VeOmni 后端进行 OPD 训练的完整配置示例，包括数据路径、模型设置（学生模型 Qwen2.5-0.5B 和教师模型 Qwen2.5-3B-Instruct）、蒸馏参数（如损失模式、批次大小）和资源分配（学生 2 GPU、教师 4 GPU）。脚本遵循 model_engine=veomni 的配置模式，并设置了 actor.veomni.* 相关参数。

4. 测试与验证：PR 描述中提到已使用 GSM8K 数据集和 Qwen 模型在 2+4 GPU 上进行了手动验证，但未包含自动化测试文件变更。作者解释这是因为该特性是集成级功能，需要多 GPU 和外部依赖，因此通过脚本验证进行端到端测试。

关键文件：

- verl/experimental/separation/engine_workers.py（模块 分离工作器；类别 source；类型 core-logic；符号 init, _get_strategy_handlers, save_model_to_cpu, restore_model_from_cpu）：核心变更文件，扩展了分离工作器以支持 VeOmni 策略，涉及模型保存 / 恢复逻辑和构造函数签名对齐。
- verl/trainer/distillation/losses.py（模块 蒸馏损失；类别 source；类型 core-logic；符号 compute_topk_loss）：关键适配文件，修改损失计算函数以支持 VeOmni 策略复用 FSDP 的损失计算逻辑。
- examples/on_policy_distillation_trainer/run_qwen_gsm8k_veomni.sh（模块 示例脚本；类别 other；类型 configuration）：新增的示例脚本，演示如何使用 VeOmni 后端进行 OPD 训练，提供完整配置参考。

关键符号：init, _get_strategy_handlers, compute_topk_loss

关键源码片段

verl/experimental/separation/engine_workers.py

核心变更文件，扩展了分离工作器以支持 VeOmni 策略，涉及模型保存 / 恢复逻辑和构造函数签名对齐。

```
class DetachActorWorker(ActorRolloutRefWorker):
    """
    扩展ActorRolloutRefWorker以支持分离和恢复演员模型的工作器类。
    当前支持FSDP、FSDP2、VeOmni和Megatron策略。
    """

    def __init__(
        self, config: DictConfig, role: str, distillation_config: Optional[DistillationConfig] = None,
        **kwargs
    ):
        """
        初始化DetachActorWorker。
        Args:
            config: 配置字典。
            role: 工作器角色（如'actor'、'rollout'、'ref'）。
            distillation_config: 可选的蒸馏配置，用于OPD支持。
            **kwargs: 传递给ActorRolloutRefWorker的额外参数。
        """
        # 将参数传递给父类，确保兼容性，特别是 distillation_config 的传递
        ActorRolloutRefWorker.__init__(self, config, role, distillation_config=distillation_config,
        **kwargs)
        self._strategy_handlers = None

    def _get_strategy_handlers(self):
```

```
        """
        获取策略特定的模型保存和恢复处理器。
        返回: (save_handler, restore_handler) 元组。
        抛出: NotImplementedError 如果策略不支持。
        """
        if self._strategy_handlers is not None:
            return self._strategy_handlers

        strategy = self.config.actor.strategy

        # VeOmni 内部使用 FSDP2 进行数据并行（VeOmniEngine 继承自 FSDPEngine 并设置 data_
        parallel_mode="fsdp2"），
        # 因此其模型参数是 DTensors，与 FSDP2 的分片保存 / 加载工具兼容。
        # 注意：当 VeOmni 的 param_offload=True 时，参数在保存 / 恢复时可能驻留在 CPU 上。
        # 当前的 fsdp2_sharded_save_to_cpu / fsdp2_sharded_load_from_cpu 假设参数在 GPU
        上。
        # 调用者应在 offload 场景中确保模型在调用 save_model_to_cpu / restore_model_from_cpu
        前已重新加载到 GPU。
        if strategy in ["fsdp", "fsdp2", "veomni"]:
            from verl.utils.fsdp_utils import (
                fsdp2_sharded_load_from_cpu,
                fsdp2_sharded_save_to_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
```

评论区精华

1. 构造函数签名兼容性争议：gemini-code-assist[bot] 指出 DetachActorWorker.__init__ 应显式包含 distillation_config 参数，以避免当参数按位置传递时（这是代码库中工作器初始化的常见模式）可能引发的 TypeError。hjshi84 解释称此变更是签名对齐修复——父类 ActorRolloutRefWorker.__init__ 已接受 distillation_config 和 **kwargs，而之前的子类构造函数因签名较窄而静默丢弃了这些参数。最终，PR 采纳了显式定义 distillation_config 参数的建议，以确保兼容性。
2. 功能范围澄清：wuxibin89 评论“We haven't support distillation in fully async training yet.”，可能误解此 PR 旨在为完全异步训练启用蒸馏。hjshi84 澄清此变更与完全异步训练无关，仅是签名对齐修复，使子类能正确将参数传递给父类。

3. 实现细节确认：wuxibin89 询问“I think pass **kwargs should be enough?”，hjshi84 回应称仅使用 **kwargs 可能足够，但遵循 gemini-code-assist[bot] 的建议显式定义 distillation_config 可防止位置参数传递问题。最终实现同时包含了 distillation_config 和 **kwargs。

- DetachActorWorker 构造函数签名兼容性 (design): 采纳建议，在构造函数中显式添加 distillation_config 参数，同时保留 **kwargs。
- 功能范围澄清 (question): 确认 PR 范围仅限于 VeOmni OPD 支持，与完全异步训练无关。

风险与影响

- 风险：
  1. 设备不匹配风险：在 verl/experimental/separation/engine_workers.py 中，新增的注释指出，当 VeOmni 的 param_offload=True 时，模型参数可能驻留在 CPU 上，而 fsdp2_sharded_save_to_cpu / fsdp2_sharded_load_from_cpu 工具假设参数在 GPU 上。如果调用者在保存 / 恢复前未确保模型已重新加载到 GPU，可能导致设备不匹配错误或数据损坏。
  2. 兼容性风险：对 DetachActorWorker 构造函数的修改（新增 distillation_config 参数）可能影响现有调用代码，如果这些代码按位置传递了额外参数而未更新，可能引发 TypeError。但根据讨论，此变更是为了修复先前因签名不匹配而静默丢弃参数的问题，因此实际风险较低。
  3. 测试覆盖不足：PR 未包含自动化测试文件变更，依赖手动脚本验证。这增加了回归风险，尤其是在 VeOmni 与 FSDP2 的集成边界，如策略处理逻辑或损失计算路径。
  4. 配置复杂性：新增的示例脚本包含大量配置参数，如果用户错误配置（如资源分配不匹配），可能导致运行时错误或性能下降。
- 影响：
  1. 对用户的影响：为 OPD 训练新增了 VeOmni 后端选项，用户现在可以使用 VeOmni 引擎进行蒸馏训练，可能受益于其特定的优化特性（如参数卸载）。示例脚本提供了快速上手的参考，降低了使用门槛。
  2. 对系统的影响：扩展了训练后端的兼容性，使系统能支持更多硬件和优化策略。对核心训练逻辑的修改较小（仅策略匹配和损失计算适配），影响范围有限，主要影响分离工作器和蒸馏损失计算模块。
  3. 对团队的影响：展示了如何将新引擎集成到现有 OPD 框架中，为未来集成其他引擎（如 TRT-LLM）提供了参考模式。代码变更集中在策略抽象层，保持了模块化设计。 - 风险标记：设备不匹配风险，缺少测试覆盖，配置复杂性

关联脉络

- PR #6051 [trainer,cfg,rollout,algo] feat: Multi-Teacher OPD: 同样涉及在线策略蒸馏（OPD）功能扩展，本 PR 的 VeOmni 支持可视为对 OPD 后端的补充。
- PR #6061 [veomni] feat: support Qwen3.5 SP and add GRPO trainer demo using VeOmniEngine: 同为 VeOmni 引擎相关 PR，扩展了 VeOmni 的模型支持和训练示例，本 PR 在此基础上新增 OPD 支持。

- PR #5997 [trainer,algo] feat: Support On-Policy Distillation in main_ppo_sync: 在同步 PPO 训练器中新增 OPD 支持，本 PR 的 VeOmni 集成可能依赖其奠定的蒸馏框架。