PR #6200 完整报告

verl-project/verl

[BREAKING][misc] refactor: migrate Diffusion RL stack to verl-omni

合并时间：2026-04-29 13:49

原文链接：http://prhub.com.cn/verl-project/verl/pull/6200

执行摘要

- 一句话：迁移 Diffusion RL 栈至独立仓库 (verl-omni)
- 推荐动作：建议读者仔细阅读 PR 的变更集，了解如何将实验性功能模块化并分离到独立仓库。对于想要维护类似解耦操作的团队，此 PR 提供了清晰的参照。

功能与动机

根据 PR body：*Move all Diffusion RL functionality out of verl-project/verl to the dedicated verl-omni repository so that Diffusion RL can iterate quickly without affecting verl's production-grade LLM training paths.*

实现拆解

1. 删除核心训练器：verl/trainer/diffusion/、verl/trainer/main_flowgrpo.py 以及所有 diffusion 配置。
2. 删除模型和引擎：verl/models/diffusers_model/、verl/workers/engine/fsdp/diffusers_impl.py、相关 rollout backend vllm_omni_async_server.py 和 agent loop diffusion_agent_loop.py。
3. 清理工具模块：删除 verl/utils/vllm_omni/、verl/utils/reward_score/jpeg_compressibility.py、verl/experimental/reward_loop/reward_manager/visual.py 等。
4. 修剪共享模块：从 workers/config、trainer/config/algorithm.py、workers/utils/losses.py 等文件中移除 diffusion-only 分支和类型定义。
5. 更新外围文件：删除 .github/workflows/vllm_omni.yml、更新 CI 允许列表、PR 模板、文档和配置生成脚本。
6. 测试文件同步删除：移除所有对应的单元测试和集成测试。

关键文件：

- verl/workers/engine/fsdp/diffusers_impl.py（模块 引擎；类别 source；类型 deletion；符号 DiffusersFSDPEngine, init, initialize, _init_device_mesh）：核心引擎实现，提供 DiffusersFSDPEngine，支持 FSDP 分片、卸载和 LoRA。删除后完全迁移至 verl-omni。
- *examples/flowgrpo_trainer/scheduler/scheduling_flow_match_sde_discrete.py*（模块 调度器；类别 source；类型 deletion；符号 FlowMatchSDEDiscreteSchedulerOutput, FlowMatchSDEDiscreteScheduler, step, sample_previous_step）：自定义 SDE 调度器

，用于 FlowGRPO 训练中带 log-prob 的扩散过程。

- verl/models/diffusers_model/base.py（模块 模型基类；类别 source；类型 deletion；符号 DiffusionModelBase, QwenImage, register, decorator）：DiffusionModelBase 抽象基类，提供注册机制和接口定义。
- verl/trainer/diffusion/ray_diffusion_trainer.py（模块 训练器；类别 source；类型 deletion；符号 RayFlowGRPOTrainer, init, _create_dataloader, _dump_generations）：FlowGRPO 训练器的 Ray 实现，包含训练循环、数据加载、指标计算等。
- examples/flowgrpo_trainer/vllm_omni/pipeline_qwenimage.py（模块 流水线；类别 source；类型 deletion；符号 _maybe_to_cpu, _coalesce_not_none, QwenImagePipelineWithLogProb, init）：QwenImage 自定义 pipeline，包装 vllm-omni 以支持扩散 log-prob 返回。
- verl/experimental/agent_loop/diffusion_agent_loop.py（模块 Agent 循环；类别 source；类型 deletion；符号 DiffusionAgentLoopOutput, _InternalDiffusionAgentLoopOutput, DiffusionAgentLoopWorker, init）：Diffusion 代理循环实现，支持多轮交互和奖励计算。
- verl/workers/rollout/vllm_rollout/vllm_omni_async_server.py（模块 Rollout 服务器；类别 source；类型 deletion；符号 vLLMOmniHttpServer, _init_model_config, _validate_configs, _post_init）：vLLM-Omni HTTP 异步服务器，处理 diffusion 推理请求。
- verl/trainer/main_flowgrpo.py（模块 入口点；类别 source；类型 deletion；符号 main, run_flowgrpo, TaskRunner, init）：FlowGRPO 训练入口点，协调资源池、工作线程和训练循环。

关键符号：DiffusersFSDPEngine.init, DiffusersFSDPEngine.initialize, FlowMatchSDEDiscreteScheduler.step, DiffusionModelBase.register, DiffusionModelBase.get_class, RayFlowGRPOTrainer.init, QwenImagePipelineWithLogProb.diffuse, DiffusionAgentLoopWorker.generate_sequences, vLLMOmniHttpServer._post_init, main_flowgrpo.run_flowgrpo

关键源码片段

verl/workers/engine/fsdp/diffusers_impl.py

核心引擎实现，提供 DiffusersFSDPEngine，支持 FSDP 分片、卸载和 LoRA。删除后完全迁移至 verl-omni。

```
# DiffusersFSDPEngine 是 diffusion 模型的 FSDP 引擎实现，
# 支持模型分片、激活 / 优化器卸载、LoRA 和序列并行。
# 此文件已删除，功能迁移至 verl-omni 仓库。
@EngineRegistry.register(model_type="diffusion_model", backend=["fsdp", "fsdp2"], device=
["cuda"])
class DiffusersFSDPEngine(BaseEngine):
    """
    Concrete Diffusers Engine implementation using PyTorch FullyShardedDataParallel (FSDP).
    Supports model sharding, activation/optimizer offloading, LoRA, and sequence parallelism.
    """
    def __init__(
```

```
        self,
        model_config: DiffusionModelConfig,
        engine_config: FSDPEngineConfig,
        optimizer_config: FSDPOptimizerConfig,
        checkpoint_config: CheckpointConfig,
    ):
        super().__init__()
        self.model_config = model_config
        self.engine_config = engine_config
        self.optimizer_config = optimizer_config
        self.checkpoint_config = checkpoint_config
        # 初始化设备网格，配置 FSDP 参数
        self._init_device_mesh()
        if self.engine_config.full_determinism:
            enable_full_determinism(seed=self.engine_config.seed)
        # 设置 offload 策略
        self._is_offload_param = self.engine_config.param_offload
        self._is_offload_optimizer = self.engine_config.optimizer_offload
        self._is_lora = self.model_config.lora_rank > 0
```

examples/flowgrpo_trainer/scheduler/scheduling_flow_match_sde_discrete.py

自定义 SDE 调度器，用于 FlowGRPO 训练中带 log-prob 的扩散过程。

```
# 自定义 SDE 调度器，用于 FlowGRPO 训练中的扩散过程，
# 基于 FlowMatchEulerDiscreteScheduler，支持 log-prob 计算。
@dataclass
class FlowMatchSDEDiscreteSchedulerOutput(BaseOutput):
    prev_sample: torch.FloatTensor
    log_prob: Optional[torch.FloatTensor]
    prev_sample_mean: torch.FloatTensor
    std_dev_t: torch.FloatTensor

class FlowMatchSDEDiscreteScheduler(FlowMatchEulerDiscreteScheduler):
    """SDE version of the FlowMatchEulerDiscreteScheduler,
       implemented for FlowGRPO (https://arxiv.org/abs/2505.05470)."""
    def step(
        self,
        model_output: torch.FloatTensor,
        timestep: float | torch.FloatTensor,
        sample: torch.FloatTensor,
        s_churn: float = 0.0,
        s_tmin: float = 0.0,
        s_tmax: float = float("inf"),
        s_noise: float = 1.0,
        generator: Optional[torch.Generator] = None,
        per_token_timesteps: Optional[torch.Tensor] = None,
        return_dict: bool = True,
        noise_level: float = 0.7,
```

```
        prev_sample: Optional[torch.FloatTensor] = None,
        sde_type: Literal["sde", "cps"] = "sde",
        return_logprobs: bool = True,
    ) -> FlowMatchSDEDiscreteSchedulerOutput | tuple:
        """反转 SDE 预测前一时刻样本，同时计算 log 概率."""
        # 实现细节省略
```

评论区精华

PR 未产生实质性讨论。只有一个自动代码审查评论（gemini-code-assist）无具体反馈，以及审核者 wuxibin89 的批准。

- 代码迁移 Review (other): 无变更请求，PR 被批准并合并。

风险与影响

- 风险：主要风险如下：
  - 兼容性破坏：现有使用 diffusion RL API 的用户必须迁移到 verl-omni，可能中断工作流程。
  - 修剪遗漏：共享模块中可能残留对已删除模块的引用，导致导入错误。需确认 workers/config、losses.py 等文件中的 conditional import 已全部清理。
  - 文档与配置：用户手册和示例仍需指向 verl-omni，否则可能误导用户。
  - 对其他引擎影响：删除 diffusion-only 分支后，共享的基础设施（如 BaseEngine、rollout 基类）需要验证没有遗漏。
  - 影响：用户：使用 diffusion RL 的用户需迁移至 verl-omni 仓库；纯 LLM 用户无影响。团队：简化主仓库维护负担，diffusion RL 可独立演进。 系统：代码库体积减小约 7700 行，CI 移除一个工作流，构建速度可能略有提升。
- 风险标记：破坏性变更，外部依赖迁移，共享分支修剪，文档更新缺失

关联脉络

- 暂无明显关联 PR