PR #7115 完整报告

verl-project/verl

[rollout, vllm] feat: pluggable router with FQN/YAML plugins

合并时间：2026-09-01 23:08

原文链接：http://prhub.com.cn/verl-project/verl/pull/7115

执行摘要

- 一句话：将负载均衡器提取为独立可插拔路由器模块，支持 FQN/YAML 插件扩展。
- 推荐动作：本 PR 是实现 Issue #6383 的关键第一步，设计合理，实现了可插拔路由器架构，且保持了良好的向后兼容性。建议精读 router.py 中的协议定义和插件加载机制，以及 llm_server.py 中客户端 RPC 载荷的惰性过滤逻辑。这些设计决策对于扩展 verl 的路由能力具有重要价值。

功能与动机

实现拆解

1. 核心路由器模块创建：在 verl/workers/rollout/router.py 中定义 RequestLoadBalancer 协议（10 个方法）和默认实现 GlobalRequestLoadBalancer（迁移自 llm_server.py，行为不变）。新增 get_router_handle() 工厂函数，根据 router_config_path 配置决定加载默认实现还是外部插件。
2. 插件加载机制实现：在 router.py 中实现插件加载链：resolve_config_path（共享工具）→ _load_router_yaml（OmegaConf 加载，拒绝 Hydra defaults 块）→ _resolve_router_class（通过 FQN 动态导入）→ _create_plugin_extension（将整个 YAML 字典作为 router_kwargs 传入插件构造函数）。
3. 客户端 RPC 载荷重构：修改 verl/workers/rollout/llm_server.py 中的 LLMServerClient。客户端本地打包所有 generate() 参数，通过惰性查询路由器声明的 require_acquire_fields() 和 require_release_fields() 方法，仅序列化路由器声明的字段到 acquire/release RPC 调用中。_release_server 签名变更为可选接收 request_id。
4. 配置与集成更新：在 verl/workers/config/rollout.py 的 RolloutConfig 中新增 router_config_path: Optional[str] = None 配置项，并同步更新 rollout.yaml 及所有生成的 trainer 配置文件。teacher_model.py 中的 _initialize_load_balancer_handle 改为使用 get_router_handle() 工厂。
5. 工具函数迁移与兼容性：将 resolve_config_path 从 verl/experimental/agent_loop/utils.py 迁移至 verl/utils/import_utils.py，并在原位置重新导出以保持兼容。agent_loop.py 中的导入路径相应更新。
6. 测试配套：新增 tests/workers/rollout/test_router_on_cpu.py，包含 22 个测试用例，覆盖 YAML 加载、插件扩展、声明式 RPC 过滤、遗留签名兼容等。更新 test_basic_agent_loop.py 中的导入路径。

关键文件：

- verl/workers/rollout/router.py（模块 路由器；类别 source；类型 entrypoint；符号 RequestLoadBalancer, GlobalRequestLoadBalancer, get_router_handle, _create_plugin_extension）：核心新模块，定义了路由器协议、默认实现和插件加载机制。
- verl/workers/rollout/llm_server.py（模块 服务器客户端；类别 source；类型 dependency-wiring；符号 LLMServerClient.init, LLMServerClient._acquire_server, LLMServerClient._release_server, LLMServerClient.generate）：核心重构文件，移除了内联负载均衡器，调整了客户端以支持声明式 RPC 载荷过滤。
- tests/workers/rollout/test_router_on_cpu.py（模块 路由器测试；类别 test；类型 test-coverage；符号 _MockPluginLoadBalancer, TestRequireFields, TestGetRouterHandleDefault, TestGetRouterHandlePluginExtensionYaml）：新增全面的单元测试，验证插件加载、声明式 RPC 过滤和向后兼容性。
- verl/utils/import_utils.py（模块 工具函数；类别 source；类型 dependency-wiring；符号 resolve_config_path）：新增共享的配置路径解析函数 resolve_config_path，供路由器和 agent loop 共用。
- verl/experimental/teacher_loop/teacher_model.py（模块 教师模型；类别 source；类型 data-contract；符号 TeacherModelManager._initialize_load_balancer_handle）：集成新路由器工厂，使教师模型也支持插件化路由。

关键符号：get_router_handle, _create_plugin_extension, _create_global_sticky_inflight, RequestLoadBalancer.acquire_server, RequestLoadBalancer.require_acquire_fields, LLMServerClient._acquire_server, LLMServerClient._release_server

关键源码片段

verl/workers/rollout/router.py

核心新模块，定义了路由器协议、默认实现和插件加载机制。

```
# verl/workers/rollout/router.py ( 核心协议与工厂函数 )

from typing import Protocol, Any
import logging
from omegaconf import OmegaConf
from verl.utils.import_utils import load_class_from_fqn, resolve_config_path

class RequestLoadBalancer(Protocol):
    """所有路由策略必须满足的结构化接口。"""
    def require_acquire_fields(self) -> list[str]:
        """声明在 acquire_server RPC 中需要的 generate() 参数名。
        例如，KV-cache 感知路由可能需要 ['prompt_ids']；
        默认路由器返回 []，仅路由基于 request_id。"""
        ...
    def require_release_fields(self) -> list[str]:
        """声明在 release_server RPC 中需要的标识字段。
        通常仅需要 ['request_id'] 用于从 acquire 时的簿记中查找；
        默认路由器返回 []，仅基于 server_id 计数。"""
```

```
        ...
    def acquire_server(self, request_id: str, **extra) -> tuple[str, Any]:
        """为给定请求获取一个服务器。
        **extra 中仅包含 require_acquire_fields() 声明的字段。"""
        ...
    def release_server(self, server_id: str, request_id: str | None = None) -> None:
        """请求完成后释放服务器。
        内容感知路由器可通过 request_id 从自身的 acquire 时簿记中查找 prompt 长度。"""
        ...
    # ... 其他方法如 add_servers, remove_servers, get_all_servers, get_status, clear_sticky_
    cache, get_total_inflight

def get_router_handle(
    servers: dict[str, Any],
    router_config_path: str | None = None,
    load_balancer_cls: type | None = None,
    full_determinism: bool = False,
) -> Any:
    """工厂函数：根据配置创建路由器 Ray Actor。
    优先级：1) 显式传入的 load_balancer_cls；2) router_config_path 指定的插件；3) 默认
    GlobalRequestLoadBalancer。"""
    if load_balancer_cls is not None:
        # 显式指定类，直接实例化（例如，verl-omni 子类）
        return ray.remote(load_balancer_cls).remote(servers=servers)
    if router_config_path:
        return _create_plugin_extension(servers, router_config_path)
    return _create_global_sticky_inflight(servers, full_determinism=full_determinism)

def _create_plugin_extension(servers: dict, config_path: str):
    """从 YAML 文件加载并实例化外部路由器插件。"""
    config_path = resolve_config_path(config_path)
    config = OmegaConf.load(config_path)
    config_dict = OmegaConf.to_container(config, resolve=True)
    router_class_fqn = config_dict.pop("router_class")
    router_cls = load_class_from_fqn(router_class_fqn)
    return ray.remote(router_cls).remote(servers=servers, router_kwargs=config_dict)
```

verl/workers/rollout/llm_server.py

核心重构文件，移除了内联负载均衡器，调整了客户端以支持声明式 RPC 载荷过滤。

```
# verl/workers/rollout/llm_server.py ( 客户端 RPC 载荷重构 )

class LLMServerClient:
    def __init__(self, config, load_balancer_handle=None, **kwargs):
        self.config = config
        self._load_balancer = load_balancer_handle
        # 惰性缓存路由器声明的字段列表
        self._lb_require_acquire_fields: list[str] | None = None
        self._lb_require_release_fields: list[str] | None = None
```

```
async def _acquire_server(self, request_id: str, **extra) -> tuple[str, ray.actor.ActorHandle]:
    """原子获取服务器，仅序列化路由器声明的字段。"""
    if self._lb_require_acquire_fields is None:
        # 首次调用时，并发查询路由器声明的字段，结果被缓存
        acquire_fields, release_fields = await asyncio.gather(
            self._load_balancer.require_acquire_fields.remote(),
            self._load_balancer.require_release_fields.remote(),
        )
        self._lb_require_acquire_fields = list(acquire_fields)
        self._lb_require_release_fields = list(release_fields)
    # 从本地打包的 extra 中筛选出路由器声明的字段，用于 RPC
    fields = {name: extra[name] for name in self._lb_require_acquire_fields if name in extra}
    return await self._load_balancer.acquire_server.remote(request_id=request_id, **fields)

def _release_server(self, server_id: str, request_id: str | None = None) -> None:
    """释放服务器，fire-and-forget。仅序列化声明的 release 字段。"""
    pool = {"request_id": request_id}
    fields = {name: pool[name] for name in self._lb_require_release_fields if name in pool}
    self._load_balancer.release_server.remote(server_id=server_id, **fields)

@rollout_trace_op
async def generate(self, request_id, *, prompt_ids, sampling_params, ...):
    """生成请求。所有参数本地打包，但仅部分字段被序列化到路由器 RPC。"""
    server_id, server = await self._acquire_server(
        request_id,
        prompt_ids=prompt_ids, # 可能被过滤
        sampling_params=sampling_params, # 可能被过滤
        ...
    )
    try:
        output = await server.generate.remote(...)
        return output
    finally:
        self._release_server(server_id, request_id=request_id) # request_id 可能被过滤
```

评论区精华

1. get_kv_events_endpoints 的必要性与设计：审核者 wuxibin89 质疑其用途，touch869 解释它是供外部路由器（如 uni-agent）获取每个副本的 kv-events ZMQ 端点以进行订阅。提出了三种设计方案，最终保持独立 getter。
2. 端口分配策略：wuxibin89 建议使用 get_free_port()，但 touch869 指出由于 vLLM 内部对端口按数据并行等级进行偏移，get_free_port() 无法感知此偏移，会导致端口冲突，因此必须使用确定性的端口分配方案（_assign_kv_events_ports）。
3. 参数命名讨论：wuxibin89 建议将 prompt_ids 重命名为 session_ids，touch869 最初同意并重命名，但 wuxibin89 随后指出 prompt_ids 是 tokenized prompt，更准确，应恢复为 prompt_ids。

4. 依赖性变更要求：wuxibin89 指出本 PR 需要包含 PR #7613（随机化 least-loaded tie-break）的更改，以避免所有会话雪崩到单一副本。

- get_kv_events_endpoints 的必要性与设计 (design): 保持独立 getter（Option 1），因为它是外部插件所需的接口，且与 rollout 配置分离。
- 端口分配策略：get_free_port 与确定性分配 (correctness): 必须使用确定性的端口分配方案（_assign_kv_events_ports），因为 get_free_port() 与 vLLM 的内部偏移机制不兼容。
- 参数命名：prompt_ids vs session_ids (style): 恢复为 prompt_ids，因为它是已 token 化的 prompt 序列，比 session_id 更精确。
- 需要包含 PR #7613 的更改 (other): 本 PR 应包含 PR #7613 的更改。

风险与影响

- 风险：向后兼容性风险：重构涉及客户端 RPC 载荷格式变更，但默认路由器声明 []，保持了 wire format 不变，且测试验证了遗留子类兼容性。 插件安全风险：通过 FQN 动态加载外部类，存在潜在的代码注入风险。缓解措施包括：配置路径解析限定在项目根目录内，YAML 配置拒绝 Hydra defaults 块。 性能风险：声明式 RPC 载荷过滤在首次调用时增加两次轻量级 RPC（查询声明），但结果被缓存，后续调用无开销。仅序列化声明字段减少了长上下文场景下的序列化开销。 配置复杂性风险：新增 router_config_path 配置项，但默认为 null，不影响现有用户。文档已更新。 测试覆盖风险：新增测试全面覆盖核心逻辑，但端到端测试未直接验证外部插件的实际路由效果。
- 影响：用户影响：用户无感知，除非主动配置 router_config_path。现有配置和行为完全不变。 系统影响：解耦了路由器与服务器管理，为系统引入新的扩展点。路由策略变更可能影响 rollout 吞吐和延迟。 团队影响：为后续实现高级路由策略（如 KV-cache 感知路由）提供了标准接口和基础设施，降低了团队成员开发新路由策略的门槛。 API 影响：LLMServerClient._acquire_server 和 _release_server 签名扩展，但向后兼容。新增 get_rollout_config() RPC 端点。
- 风险标记：plugin-security, backward-compatibility, rpc-serialization-change

关联脉络

- PR #6383 [RFC] KV-cache-aware request load balancer for rollout servers: 关联的 Issue，提出了可插拔路由器的需求和设计目标，是本 PR 的动机来源。
- PR #7613 [rollout] fix: randomize least-loaded tie-break so sessions do not avalanche onto one replica: 被审核者在讨论中明确指出需要包含此 PR 的更改（随机化 tie-break 以避免会话雪崩），属于功能依赖。
- PR #7227 [ckpt, rollout, vllm] feat: add vLLM consumer for delta-sharded weight sync: 同属 vLLM 路由 / 通信相关变更，涉及服务器管理和权重同步，可能共享基础设施。
- PR #7630 [rollout] fix: DeepSeek continuous token builder cannot render tool appends: 同属 rollout 模块的修复，可能与路由器处理的 token 构建相关。