PR #5978 完整报告

verl-project/verl

[tool, rollout, cfg] feat: per-sample tool environment routing for ToolAgentLoop

合并时间：2026-04-16 16:11

原文链接：http://prhub.com.cn/verl-project/verl/pull/5978

执行摘要

- 一句话：为 ToolAgentLoop 新增基于样本的工具选择功能，支持多轮 rollout 中每个样本使用不同工具子集。
- 推荐动作：此 PR 值得精读，因为它展示了如何在现有状态机中嵌入样本级配置，而无需大规模重构。关注 run() 方法中的工具筛选逻辑和 getattr 的使用，这体现了灵活的设计决策，允许逐步迁移到更复杂的工具管理系统。同时，注意 review 中关于实现与描述不符的讨论，这提醒了保持文档同步的重要性。

功能与动机

PR body 指出，在真实世界的 agent 训练中，不同场景需要不同的工具（例如 gsm8k 计算器、sandbox_fusion 代码执行器、geo3k 几何工具）。目前，批次中的所有行共享相同的全局 tool_config_path，这限制了异构工具环境的使用。此 PR 旨在通过简单的配置映射和数据集级别的路由键，实现异构工具环境，而无需任何自定义子类或新的 agent loop。

实现拆解

1. 入口逻辑修改：在 verl/experimental/agent_loop/tool_agent_loop.py 的 run() 方法中，新增代码从 kwargs.get("extra_info", {}) 提取 tool_selection 列表。如果提供了 tool_selection 且全局工具字典 self.tools 非空，则根据列表筛选出匹配的工具子集，并赋值给 agent_data._active_tools 和 agent_data._active_tool_schemas；否则，使用全局工具作为默认值。
2. 状态机逻辑适配：修改 _handle_pending_state()、_handle_generating_state() 和 _call_tool() 方法，使用 getattr(agent_data, "_active_tools", self.tools) 或 getattr(agent_data, "_active_tool_schemas", self.tool_schemas) 来获取每个样本的活动工具，而不是直接使用全局工具。这确保了在生成提示、提取工具调用和执行工具时，都使用样本特定的工具集。
3. 配置与兼容性：未对 verl/trainer/config/rollout/rollout.yaml 进行实质性修改，仅添加了 tool_envs: null 默认值和文档说明，保持向后兼容性。当 tool_selection 未提供或为空时，行为与当前主分支一致，使用全局工具。

关键文件：

- verl/experimental/agent_loop/tool_agent_loop.py（模块 Agent 循环；类别 source；类型 core-logic；符号 run, _handle_pending_state, _handle_generating_state, _call_tool）：

这是核心实现文件，修改了 ToolAgentLoop 的状态机逻辑，以支持基于样本的工具选择。

- verl/trainer/config/rollout/rollout.yaml（模块 配置；类别 config；类型 configuration）：配置文件，添加了 tool_envs 的默认值和文档说明，但未改变核心逻辑，主要用于向后兼容和文档。
- verl/workers/config/rollout.py（模块 配置；类别 source；类型 data-contract；符号 MultiTurnConfig）：配置类文件，在 MultiTurnConfig 中添加了 tool_envs 字段，支持未来扩展，但当前实现未直接使用。

关键符号：run, _handle_pending_state, _handle_generating_state, _call_tool

关键源码片段

verl/experimental/agent_loop/tool_agent_loop.py

这是核心实现文件，修改了 ToolAgentLoop 的状态机逻辑，以支持基于样本的工具选择。

```
async def run(self, sampling_params: dict[str, Any], **kwargs) -> AgentLoopOutput:
    # ... 之前的代码创建 agent_data ...

    # Per-sample tool selection: filter global tools by extra_info.tool_selection
    extra_info = kwargs.get("extra_info", {}) or {}
    tool_selection = extra_info.get("tool_selection")
    if tool_selection and self.tools:
        # 筛选出在全局工具字典中存在的工具名称
        selected = {name: self.tools[name] for name in tool_selection if name in self.tools}
        agent_data._active_tools = selected # 存储样本特定的工具字典
        agent_data._active_tool_schemas = [
            t.tool_schema.model_dump(exclude_unset=True, exclude_none=True) for t in selected.
            values()
        ] # 生成对应的工具模式列表
    else:
        # 如果没有提供 tool_selection 或全局工具为空，则使用全局工具作为默认值
        agent_data._active_tools = self.tools
        agent_data._active_tool_schemas = self.tool_schemas

    # 状态机循环继续使用 agent_data 中的活动工具
    state = AgentState.PENDING
    while state != AgentState.TERMINATED:
        if state == AgentState.PENDING:
            state = await self._handle_pending_state(agent_data, sampling_params)
        # ... 其他状态处理 ...
```

评论区精华

review 中主要讨论了实现方案的设计权衡：

- wuxibin89 建议简化方案：提出“只需在 extra_info 中添加 tool_selection，保持 rollout.yaml 不变，从 tool_config_path 加载所有工具，然后通过 tool_selection 选择子集”，这引导了最终实现方向，避免了复杂的预加载缓存机制。

- gemini-code-assist[bot] 指出实现与 PR 描述不符：描述中提到基于 tool_env_name 和预加载缓存的环境路由，但实际代码使用 tool_selection 过滤全局工具，且未修改 __init__。这揭示了 PR 描述可能未及时更新，但实现本身更简单直接。
- 未来方向讨论：wuxibin89 提到正在开发基于注册的工具定义，以消除对 agent_loop_config_path 的依赖，此 PR 被视为一个临时解决方案，支持实例级环境指定，而不修改核心算法。
- 实现方案简化 (design): PR 作者采纳了此建议，修改代码使用 tool_selection 过滤全局工具，简化了实现。
- 实现与描述不符 (correctness): 未明确解决，但实现更简单，可能 PR 描述未及时更新；风险点在于如果 tool_selection 中名称不匹配，可能导致工具集为空。
- 未来工具定义方向 (design): 此 PR 被视为过渡方案，不修改核心算法，为未来功能铺路。

风险与影响

- 风险：1. 逻辑缺陷风险：如果 tool_selection 列表中的工具名称在全局工具中不存在，筛选后的 selected 字典可能为空，导致样本没有可用工具。当前实现中，当 tool_selection 存在但 self.tools 为空时，会回退到全局工具，但如果 self.tools 非空但无匹配名称，selected 可能为空，这可能导致后续工具调用失败。代码中未明确处理此边缘情况。 2. 兼容性风险：修改了 AgentData 对象，新增了 _active_tools 和 _active_tool_schemas 属性，如果其他代码依赖 AgentData 的结构，可能引入意外行为。但鉴于这是实验性模块，影响范围有限。 3. 性能风险：每次运行 run() 时都进行工具筛选，如果工具列表很大或调用频繁，可能增加开销，但鉴于工具数量通常有限，风险较低。
- 影响：1. 对用户的影响：用户现在可以在数据集中为每个样本指定 extra_info.tool_selection 列表，以使用不同的工具子集，这增强了训练灵活性，支持异构环境下的 agent 训练，而无需修改配置或代码。 2. 对系统的影响：仅影响 ToolAgentLoop 的实验性模块，不改变核心训练逻辑或其他模块，向后兼容性好，未配置 tool_selection 时行为不变。 3. 对团队的影响：为未来基于注册的工具定义铺平道路，提供了简单的过渡方案，团队可以立即开始使用样本级工具选择，同时等待更高级的功能。
- 风险标记：逻辑边缘情况未处理，实验性模块变更

关联脉络

- PR #5971 [reward] feat: add compute_score timing metrics to agent loop: 同属 agent_loop 实验性模块的改进，关注性能指标，而此 PR 关注工具选择功能，显示该模块的持续演进。
- PR #5988 [fully_async] feat: enable fully async to log_val_generations: 涉及 rollout 和训练器功能，与此 PR 在工具和环境配置上有间接关联，都旨在增强训练灵活性。