PR #2005 完整报告

THUDM/slime

[coding-agent-rl] Refactor coding-agent RL: turn-node TrajectoryManager + pluggable harness layer

合并时间：2026-06-17 10:08

原文链接：http://prhub.com.cn/THUDM/slime/pull/2005

执行摘要

本 PR 对 coding-agent RL rollout 子系统进行了核心重写，用基于消息树 (TrajectoryManager) 的轨迹管理器替换了原有的分段线性模型，并新增了可插拔的 harness 层（Claude Code / Codex）。整个重构涉及 27 个文件，净增加 ~4900 行，删除 ~3300 行，测试覆盖达到 98%。主要驱动因素是多轮对话中 token drift 的精确处理需求，以及支持更多编码代理的扩展性需求。

功能与动机

原有的轨迹管理基于手动分段（subagent/wipe/final），在多轮对话场景中存在以下问题：

1. Token drift：TITO (Text-In-Token-Out) 与 chat template 重标记导致 token ID 漂移，原有模型无法处理，只能抛弃整个样本。
2. 分支支持缺失：当 agent 重放会话时，可能会对历史消息进行微调（如 re-order tool call arguments），原有模型无法准确匹配，导致训练数据混淆。
3. 扩展性差：sandbox 引导、CLI 注入、配置写入等逻辑耦合在示例层，添加新的编码代理（如 Codex）需要大量重复工作。

动机直接来自实际训练中观测到的数据质量问题。PR body 明确指出："The old implementation linearized a rollout into reward-split segments; the new TrajectoryManager models a session as a per-sid message treeof turn nodes."

实现拆解

1. 重写轨迹管理层

核心文件：slime/agent/trajectory.py

- 引入 TurnRecord (prompt_ids, output_ids, finish_reason, output_log_probs) 作为适配器与管理器之间的契约。
- TrajectoryManager 基于 per-session 消息树管理多轮对话：record_turn() 将每个 turn 插入树中，通过 dict 比较前缀消息进行挂载点匹配。
- get_trajectory() 将树线性化为带 loss mask 的 Sample 列表，确保只有最新叶子的 response 参与训练。
- 支持三类 token drift 处理：
  - Fork：漂移位于 prompt 区域或叶子侧，创建新分支。

- Replace：漂移位于当前 response 内部且长度低于阈值，直接替换并设置 loss=0。
- Merge：针对 assistant 消息的轻微重写（如 whitespace 变化），合并到原始节点。
- 性能优化：使用 chunk 4096 比较 (_common_prefix_len) 和 dict == 直接比较挂载点，匹配性能提升 10-100 倍。

2. 构建可插拔 harness 层

核心文件：slime/agent/harness/common.py 及其子类

- 定义 BaseHarness 抽象基类，包含三个钩子：install_cli, write_config, launch_and_wait。
- run() 模板方法提供通用流程：ensure_agent_user -> 创建 HarnessContext -> write_config -> launch_and_wait。
- 实现 ClaudeCodeHarness 和 CodexHarness，共享以下辅助函数：
  - run_command()：处理进程分离、超时、轨迹日志捕获，返回 exit code 或超时标志。
  - install_npm_cli()：通用 npm 包 CLI 安装。
  - HarnessContext 数据类封装运行时上下文，与任务无关。

3. 重构适配器层并集成 TrajectoryManager

核心文件：slime/agent/adapters/common.py, anthropic.py, openai.py

- 将 AnthropicAdapter 和 OpenAIAdapter 的公共逻辑抽取到 BaseAdapter：
  - Session 管理每个会话的采样参数和 context budget。
  - Reply 数据类统一 _build_reply 输出。
  - _run_turn 流水线：渲染 prompt -> 调用 sglang -> 构建 reply -> 记录 turn。
  - 适配器实例持有全局 TrajectoryManager 实例，finish_session() 调用 get_trajectory() 返回训练样本。
  - 删除旧的 AdapterChain、TokenSegment 等类，简化数据结构。
  - 环境变量统一为 SLIME_AGENT_* 和 ADAPTER_*，并提供旧名称的 fallback。

4. 清理示例层并分离 SWE 逻辑

核心文件：examples/coding_agent_rl/swe.py（新增），sandbox.py（删除）

- sandbox.py 被删除，其功能拆分为：
  - slime/agent/sandbox.py：E2B sandbox 创建、agent 用户创建、文件写入基础功能。
  - examples/coding_agent_rl/swe.py：SWE 任务特定逻辑（数据集元数据提取、工作区准备、diff 捕获、评估）。
  - swe.py 定义 get_metadata() 统一两种数据集 schema，prepare_workspace() 执行 swepro setup + pre_commands + 写入 PROBLEM_STATEMENT.md。
  - 评估逻辑统一在 evaluate() 中，使用独立的评估 sandbox，支持 swepro / eval_cmd / f2p_script 三种模式。

5. 测试重组并增强 CI

- 测试文件从顶层分散 (test_agent_adapters.py, test_agent_sdk_adapters.py, test_agent_trajectory.py) 重组为 tests/test_agent/ 包。

- 新增 test_trajectory_manager_branching.py（1396 行，覆盖 98% 分支），包含 4 组测试：
  - 路由树层 (message-identity forks)
  - 线性化层 (token-id drift, dedup, reward split)
  - 结合场景 (rewrite-merge, tree-fork + token-drift, deep multi-leaf)
  - 边界 / 防御性场景 (tools metadata, logprobs mismatch, empty messages)
  - 测试采用 golden token+loss 字符串断言，确保每个样本的 token 和 loss mask 准确。
  - CI 新增 agent-test 任务，在 pr-test.yml.j2 中注册。

slime/agent/harness/common.py

新增可插拔 harness 基类和辅助函数，是架构扩展性的关键

```
"""Harness-agnostic coding-agent lifecycle in a sandbox.

A harness is a swappable coding agent (Claude Code, Codex, ...). Each one
installs a CLI, writes its own config, and runs the agent against a prompt. The
shared parts (create the agent user, the run skeleton, the launch-detached-and-
poll transport) live here; adding a CLI-style harness means subclassing
BaseHarness and implementing install_cli, write_config and launch_and_wait.
The base knows nothing about the task: run() takes only generic fields
(workdir / session_id / adapter_url / prompt). Task-specific workspace prep and
scoring live in the example layer.
"""

from __future__ import annotations

import asyncio
import lzma
import os
import shlex
import shutil
import tempfile
import time
from abc import ABC, ABCMeta, abstractmethod
from dataclasses import dataclass
from pathlib import Path

from slime.agent import sandbox as _sandbox
from slime.agent.sandbox import Sandbox
from slime.utils.misc import SingletonMeta


class SingletonABCMeta(ABCMeta, SingletonMeta):
    """Combine abstract base class (ABC) with singleton behavior."""
    pass

EXIT_TIME_BUDGET_EXCEEDED = -1 # sentinel when agent run times out
```

```
@dataclass(frozen=True)
class HarnessContext:
    """Generic run context, free of any task fields.

    ``model_label`` is the model name the harness advertises to its CLI. The
    slime adapter ignores it and serves whatever upstream sglang has loaded.
    """
    workdir: str # workspace path inside sandbox
    session_id: str # session identifier for TrajectoryManager
    adapter_url: str # URL for reverse-connection LLM adapter
    model_label: str = "slime-actor"


class BaseHarness(ABC, metaclass=SingletonABCMeta):
    """Base lifecycle for a sandbox-resident coding agent."""

    name: str = "" # short identifier, set by subclass (e.g., "claude_code")

    @abstractmethod
    async def install_cli(self, sb: Sandbox) -> None:
        """Install the harness CLI into the sandbox.
        npm-packaged harnesses delegate to install_npm_cli."""

    @abstractmethod
    async def write_config(self, sb: Sandbox, ctx: HarnessContext) -> None:
        """Write any CLI config files into the sandbox."""

    @abstractmethod
    async def launch_and_wait(self, sb: Sandbox, ctx: HarnessContext, prompt: str, time_budget_
    sec: int) -> int:
        """Run the agent to completion and return its exit code.
        A non-interactive CLI builds one shell command and hands it to
        run_command. An interactive or long-running harness drives its own loop.
        """

    async def run(
        self,
        sb: Sandbox,
        *,
        workdir: str,
        session_id: str,
        adapter_url: str,
        time_budget_sec: int,
        prompt: str,
    ) -> int:
        """Template method: ensure agent user -> write config -> launch & wait.
        Workspace prep (writing problem statement etc.) is the caller's job.
        """
        await _sandbox.ensure_agent_user(sb, workdir)
```

```
        ctx = HarnessContext(
            workdir=workdir,
            session_id=session_id,
            adapter_url=adapter_url,
        )
        await self.write_config(sb, ctx)
        return await self.launch_and_wait(sb, ctx, prompt, time_budget_sec)
```

评论区精华

Performance hotspot: _common_prefix_len

- zhuzilin: " 这个 for 循环会比较慢 ... 我不太确定这里会不会是一个瓶颈 ..."
- jingshenghang: 采用 chunk 4096 分块比较，1M 长度场景从 36ms 降至 12ms，3× 性能提升。函数名也改为更清晰的 _common_prefix_len。

Mount-point matching: dict == vs json.dumps

- zhuzilin: "dict == 能匹配内部的内容吗？是不是只能匹配 reference id"
- jingshenghang: 演示 dict == 递归匹配 key-value 但顺序不敏感，匹配耗时从 1198ms 降至 17.6ms。由于 CC 可能调整 dict 顺序，顺序变化由 token drift 探测处理。

Lock and exception handling

- zhuzilin: 询问 adapter 中 lock 和 try-catch 必要性。
- jingshenghang: 确认无并发（CC 保证顺序），删除 lock；预期无异常，删除 try-catch，允许直接失败。

Documentation style

- zhuzilin: " 我们可能需要把 docs 变得没有那么强的 ai 味 ..."
- jingshenghang: 接受并大幅精简，仅保留跨层合同、不变约定和陷阱。

Test naming

- zhuzilin: " 目前 slime 里面的 e2e 测试都是指进行训练的。不建议这里叫 e2e。"
- jingshenghang: 重命名为 test_trajectory_manager_branching.py。

CC title-generation requests

- zhuzilin: " 这是什么魔鬼逻辑。。。"
- jingshenghang: 解释是 CC 自动发起的 session title 请求，通过切换到 claude -p 启动规避并删除过滤代码。

Billing header

- zhuzilin: 询问 billing header 过滤的必要性。
- jingshenghang: 通过设置 CLAUDE_CODE_ATTRIBUTION_HEADER=0 关闭，删除代码过滤。

风险与影响

- 训练数据生成路径改变：TrajectoryManager 的 token drift 处理逻辑（fork/replace）可能对模型训练产生不可预期的影响。但通过 98% 测试覆盖和 review 中的反复打磨，风险可控。
- 环境变量迁移：旧环境变量（SWE_HOST_*, SLIME_HEAD_HOST, SHIM_*）被替换为 SLIME_AGENT_* 和 ADAPTER_*。虽然通过 _getenv 提供向后兼容，但用户仍需更新部署脚本。文档已同步更新。
- harness 单例状态：BaseHarness 采用 SingletonABCMeta，若安装或配置有残留状态可能影响后续调用。但 harness 设计为无状态（仅在运行时安装 CLI、写入配置），风险低。
- 适配器重构：旧适配器 API（如 AdapterChain）被移除，外部代码若直接使用需更新。但示例已更新反映新 API。

关联脉络

本 PR 与近期 coding_agent_rl 功能线形成直接联动：

- PR #2161：引入可配置的评分协议和 sandbox RPC 稳健性增强，与本 PR 的可插拔 harness 共享相同的 slime/agent/ 和 examples/coding_agent_rl/ 模块，二者共同推进 agentic RL 的基础架构。
- PR #2183：对 slime 其他模块进行清理，与本 PR 的清理趋势一致，体现团队对代码质量的持续关注。

此外，本 PR 的 TrajectoryManager 设计为后续的 token-faithful 训练和更复杂的 multi-turn RL 提供了标准基础设施。Issue 中社区成员也提到 "landed on almost exactly your structure: a per-session tree of turn nodes"，验证了该设计方向的正确性。未来可以基于此添加更细粒度的 drift 控制和不同协议的 adapter。