PR #2161 完整报告

THUDM/slime

feat(coding_agent_rl): env-selectable grading protocol + sandbox RPC robustness

合并时间：2026-07-02 11:05

原文链接：http://prhub.com.cn/THUDM/slime/pull/2161

执行摘要

本 PR 为 SWE coding-agent RL 添加了可环境变量配置的评分协议（scaleswe / swebench），并显著增强了 E2B sandbox 的 RPC 稳健性——通过分离式命令执行（exec_and_wait）和幂等重试机制解决了长时间任务因连接中断丢失结果的问题。同时，轨迹样本会根据 session 的 max_context_tokens 进行截断，防止训练上下文溢出。这些改动从内部同步，是一次模块化和可靠性提升。

功能与动机

随着 SWE-bench Verified 等新数据集引入，原有的单一评分逻辑（scaleswe 自定义协议）无法满足不同数据格式的需求。训练和评估路径可能需要使用不同的评分器（例如训练用 scaleswe、评估用 swebench 官方工具）。此外，E2B sandbox 的 HTTP/2 网关对长时间运行命令的流连接有限制：超过一定时间后连接被切断，导致退出码丢失，且无法安全重试非幂等操作。现有 run_command 虽然做了 detach 但仍有竞争和可靠性问题。轨迹样本目前无长度限制，过长的多轮轨迹可能产生超出训练窗口的样本，引发训练崩溃。

实现拆解

1. 评分协议分派（examples/coding_agent_rl/swe.py） - get_metadata 新增 protocol 参数，路由到 _metadata_scaleswe 或 _metadata_swebench。 - _metadata_scaleswe 保留原 scaleswe 数据形状，将评分参数打包到 grading 子字典。 - _metadata_swebench 处理 SWE-bench Verified 的 remote_env_info 结构，将整个 instance 字典传递给 make_test_spec。 - 新增 EvalResult 命名元组和 run_evaluation 函数，封装评分流程；evaluability_check 在生成前快速校验实例是否可评。
2. Sandbox RPC 稳健性（slime/agent/sandbox.py） - exec_and_wait：写出启动器脚本 → setsid 后台执行 → 轮询退出码标记文件（_await_done_marker）。 - _is_transient_rpc_error：判断是否暂态错误（如 h2 协议错误、SSL 错误），对等幂操作执行带抖动退避的重试。 - _rpc_retry 和 _reset_conn_pool：在 sb.exec 外层包裹重试逻辑，遇到协议错误时重置连接池。 - 引入 EXIT_TIME_BUDGET_EXCEEDED = -1，统一超时退出码。
3. Harness 适配（slime/agent/harness/common.py） - run_command 被 run_agent 取代，内部直接调用 sandbox.exec_and_wait，简化了 shell 拼接和文件管理。 - install_npm_cli 增加 NPM_INSTALL_RETRIES 和 NPM_INSTALL_BACKOFF_SEC，处理 npm 全局安装的暂态故障（如 exit 217）。

4. 轨迹截断（slime/agent/trajectory.py、slime/agent/adapters/common.py） - _SampleBuilder.to_sample 新增 max_sample_tokens 参数：对 tokens、loss_mask、rollout_log_probs 统一从开头截断。 - get_trajectory 和 _chain_to_samples 逐层透传 max_sample_tokens，该值在 finish_session 中来自 session 的 max_context_tokens。 - ill_formed 标记：TurnRecord 新增 ill_formed 布尔字段，用于标识解析异常的回合。
5. 生成器入口（examples/coding_agent_rl/generate.py） - SweConfig 新增 eval_protocol 和 train_protocol，从环境变量 SWE_EVAL_PROTOCOL / SWE_TRAIN_PROTOCOL 读取，默认 scaleswe。 - generate() 新增 evaluation: bool = False 参数，据此选择对应协议。 - 在 boot_agent_sandbox 的退避中增加随机抖动（random.random()），避免启动雪崩。
6. 启动脚本（examples/coding_agent_rl/run_qwen36_35b_a3b_swe_8nodes.sh） - 传递 SWE_TRAIN_PROTOCOL 给 Ray worker。
7. 测试同步（tests/test_agent/） - 测试用例更新以匹配 run_command → run_agent 的重命名及 exec_and_wait 的接口变更。

swe.py：评分协议分派

```
def get_metadata(sample: Sample, protocol: str = PROTOCOL_SCALESWE) -> dict[str, Any]:
    """根据 protocol 选择数据集元数据解析器，返回统一的 md 字典。

    scaleswe 协议使用自定义评分（exit 0 == 通过），swebench 协议
    利用官方 make_test_spec + get_eval_report，支持每个 repo 自己的测试命令。
    """
    if protocol == PROTOCOL_SWEBENCH:
        return _metadata_swebench(sample)
    return _metadata_scaleswe(sample)


def _metadata_scaleswe(sample: Sample) -> dict[str, Any]:
    """scaleswe 数据格式：flat metadata.* + remote_env_info 回退。

    grading 字段承载 swepro、eval_cmd、f2p_script 等评分所需信息。
    """
    m = sample.metadata or {}
    rem = m.get("remote_env_info") or {}
    label = sample.label if (isinstance(sample.label, str) and len(sample.label) < 256) else None
    swepro = m.get("swepro")
    eval_cmd = m.get("eval_cmd")
    f2p_script = rem.get("f2p_script")
    looks_swebench = bool(rem.get("test_patch")) and not (swepro or eval_cmd or f2p_script)
    return {
        "protocol": PROTOCOL_SCALESWE,
        "instance_id": m.get("instance_id") or rem.get("instance_id") or label or "unknown",
        "image": m.get("image") or rem.get("image_url"),
        "workdir": m.get("workdir") or rem.get("workdir"),
        "problem_statement": m.get("problem_statement") or _coerce_prompt(sample.prompt),
        "looks_swebench": looks_swebench,
```

```
        "grading": {
            "swepro": swepro,
            "eval_cmd": eval_cmd,
            "f2p_script": f2p_script,
            "pre_commands": m.get("pre_commands") or rem.get("pre_commands"),
        },
    }
```

sandbox.py：分离式命令执行

```
async def exec_and_wait(
    sb: Sandbox, *, cmd: str, time_budget_sec: int, tag: str,
    user: str = "root", env: dict[str, str] | None = None, workdir: str | None = None,
    out_file: str | None = None, want_output: bool = False,
) -> tuple[int, str]:
    """分离式执行命令，适合长时间任务（如测试套件）。

    原理：写一个启动器脚本 setsid 到后台，将输出重定向到文件，
    然后通过轮询标记文件获取退出码。这样即使 E2B 网关断开流连接，
    也不会丢失结果。轮询同时作为 keepalive 防止 sandbox 被空闲回收。
    """
    out_file = out_file or f"/tmp/.{tag}.out"
    done_file = f"/tmp/.{tag}.done"
    launcher = f"/tmp/.{tag}.sh"
    lock_dir = f"/tmp/.{tag}.spawned"
    prefix = f"cd {workdir}\nexport HOME=/home/{user}\n" if workdir else ""
    launcher_body = f"#!/bin/bash\n{prefix}{cmd}\necho $? > {done_file}\n"
    await sb.write_file(launcher, launcher_body, user=user)
    await sb.exec(
        f"chmod +x {launcher}; "
        f"mkdir {lock_dir} 2>/dev/null || exit 0; "
        f"rm -f {out_file} {done_file}; "
        f"setsid bash {launcher} < /dev/null > {out_file} 2>&1 &",
        user=user, env=env, timeout=30, check=True, idempotent=True,
    )
    exit_code = await _await_done_marker(sb, done_file, user=user, time_budget_sec=time_
    budget_sec)
    if exit_code == 0 and not want_output:
        return exit_code, ""
    if want_output:
        return exit_code, await sb.read_file(out_file, user=user)
    _, tail, _ = await sb.exec(f"tail -c 512 {out_file} 2>/dev/null", user=user, timeout=15, check=
    False)
    return exit_code, tail or ""
```

评论区精华

无 review 讨论。

风险与影响

- 幂等性假设风险（sandbox.py）：_is_transient_rpc_error 判断可能遗漏异常类型，导致非幂等命令被错误重试。实际中可通过日志监控紧急处理。
- 协议兼容性风险（swe.py）：_metadata_swebench 在 swebench 包未安装时静默失败，导入异常被捕获但不会阻止程序启动，直到实际调用 run_evaluation 时才会暴露。建议在启动时检查包可用性。
- 轨迹截断风险（trajectory.py）：截断时假定 tokens 按原始顺序，若 leading_prompt_len 计算错误（如 ill_formed 回合）会导致 loss_mask 偏移，训练标签错误。单元测试覆盖了基本截断场景。
- 并发文件锁风险：exec_and_wait 用 mkdir lock_dir 作为锁，但若多个协程并发使用相同 tag 可能只有第一个成功，其余静默跳过。当前 tag 由调用方保证唯一（如 "run" 固定），长期建议改用 sandbox 内部基于 random 的标记。

影响范围限于 coding_agent_rl 示例和 slime/agent 模块，不影响其他 backends 或框架。团队获得更灵活的评分配置和更稳健的 sandbox 运行环境。

关联脉络

本 PR 与 #2125（SWE_AGENT 环境变量选择 agent）构成系列：一个选择 agent，一个选择评分协议，共同支持 configurable RL pipeline。#2124 修复了 agent 的 abort 处理和 session 清理，而本 PR 从 Sandbox 层面系统性解决了因连接断开导致的丢失结果问题。#2118 也是内部同步，但聚焦 Megatron 后端；本 PR 则完全在 agent 层，两者互补。未来可以在此基础上统一 agent 的配置入口，或者将评分协议扩展为插件化。