PR #2239 完整报告

THUDM/slime

fix: clear exec_and_wait's spawn lock between logical invocations

合并时间：2026-08-12 13:46

原文链接：http://prhub.com.cn/THUDM/slime/pull/2239

执行摘要

- 一句话：修复 exec_and_wait 同 tag 二次调用静默复用旧结果的问题
- 推荐动作：该 PR 值得精读，尤其是根因分析和修复取舍：mkdir guard 的传输级去重与逻辑调用级清理如何解耦。对于依赖 exec_and_wait 的 agent 代码，建议检查是否有同 tag 重叠调用的隐患，并确认第三方 sandbox 实现已接受 idempotent 参数。测试中的 ShellFakeSandbox 设计（解释 shell 字符串而非逐命令 mock）也值得借鉴。

功能与动机

PR body 明确指出：exec_and_wait 的 retry protection 让重试变得不可能，第二次使用相同 tag 的调用什么都不执行、静默返回第一次调用的退出码和日志。现实受害者是 harness.common.install_npm_cli，它设计上会用 tag="harness-npm-install" 重试 npm install 三次（NPM_INSTALL_RETRIES = 3），第 2、3 次尝试实际没有执行任何命令，只是重放了第 1 次的结果，导致本可在重试中成功的瞬时磁盘抖动变成硬失败。

实现拆解

该变更围绕 slime/agent/sandbox.py 的 exec_and_wait 展开，共分四步：

1. 核心修复：把清理动作从 guarded spawn 中拆出。原命令 chmod +x {launcher}; mkdir {lock_dir} 2>/dev/null || exit 0; rm -f {out_file} {done_file}; setsid bash ... 中，rm -f 清理位于 guard 之后，一旦 guard 命中则旧 marker 残留。改为在 spawn 之前单独发送一个幂等清理 RPC：rm -rf {lock_dir}; rm -f {out_file} {done_file}，保证每个逻辑调用开始时状态干净；mkdir guard 不再承担跨调用职责，只对本次 spawn RPC 的传输层重放去重。
2. 协议修正：Sandbox.exec 增加 idempotent: bool = True。此前 exec_and_wait 已经在两次 sb.exec 调用中传了 idempotent，但 Sandbox Protocol 没有声明该参数，任何严格按文档签名实现的第三方 sandbox 都会在调用时抛 TypeError。本 PR 将参数补入签名，并更新 docstring 说明它是后端传输重试策略的提示，无重试的后端可忽略。
3. 语义级回归测试：新增 tests/test_agent/test_sandbox_exec_and_wait.py，ShellFakeSandbox 通过正则解析 exec_and_wait 实际发出的 shell 字符串（mkdir guard 短路、rm 清理、setsid 启动、marker 轮询），不依赖具体命令布局。包含两个核心用例：test_same_tag_reinvocation_actually_reruns 验证同 tag 重新调用会真正执行（在 main 上会失败并返回上一次的 (1, "attempt-1 failed")）；replayed spawn RPC 仍被去重，且清理动作不在 guarded 命令内部。

4. CI 接入：在 .github/workflows/pr-test.yml.j2 及生成的 pr-test.yml 的 agent-test job 中加入test_agent/test_sandbox_exec_and_wait.py，num_gpus:0，作为纯CPU测试运行。

关键文件：

- slime/agent/sandbox.py（模块 沙箱抽象；类别 source；类型 core-logic；符号 exec_and_wait, Sandbox, _await_done_marker）：核心修复文件：调整 exec_and_wait 的 spawn 命令结构，将清理动作前置到独立幂等 RPC，并给 Sandbox.exec Protocol 补上 idempotent 参数，消除签名不一致。
- tests/test_agent/test_sandbox_exec_and_wait.py（模块 沙箱测试；类别 test；类型 test-coverage；符号 ShellFakeSandbox, init, aenter, aexit）：新增语义级回归测试，通过 ShellFakeSandbox 解释 exec_and_wait 实际发出的 shell 字符串来固定去重与清理语义，直接覆盖 main 上失败的 bug 场景，是本 PR 的验证核心。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：将新测试文件接入 agent-test CI job，确保回归测试在 CI 中持续运行。
- .github/workflows/pr-test.yml.j2（模块 CI 配置；类别 infra；类型 infrastructure）：CI 模板源文件，与生成的 pr-test.yml 同步更新。

关键符号：exec_and_wait, Sandbox.exec, ShellFakeSandbox.exec, ShellFakeSandbox._run_shell_fragment

关键源码片段

slime/agent/sandbox.py

核心修复文件：调整 exec_and_wait 的 spawn 命令结构，将清理动作前置到独立幂等 RPC，并给 Sandbox.exec Protocol 补上 idempotent 参数，消除签名不一致。

```
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]:
    """以完全分离的方式运行 cmd 到完成，返回 (exit_code, output)。

    普通 sb.exec 会为整个命令生命周期保持 HTTP/2 流打开；长时间命令会超出
    E2B gateway 对单个响应流可保持的时间，流被切断后我们会丢失退出码且无法
    安全重试非幂等命令。因此这里用 setsid 让命令完全分离，输出重定向到文件，
    退出码写入 marker 文件；调用方侧就变成一串短的、幂等的 RPC：写 launcher、
    触发 spawn、轮询 marker（见 _await_done_marker），都不依赖流保持存活，
    轮询本身也作为命令运行期间的 idle-GC keepalive。
    """
```

```
    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)

    # 在 guarded spawn 之前，用独立的幂等 RPC 清除上次调用的状态。
    # mkdir guard 只用于对这一次 spawn 的传输级重试去重（severed response
    # 被 _rpc_retry 重放时不能重复执行）；它不能泄漏到同一个 tag 的下一次
    # 逻辑调用（例如 install_npm_cli 的 retry 循环），否则会直接跳过 spawn
    # 并读到上一次运行遗留的 exit-code marker。调用方不得让同一个 tag 的
    # 两次 exec_and_wait 重叠执行。
    await sb.exec(
        f"rm -rf {lock_dir}; rm -f {out_file} {done_file}",
        user=user,
        timeout=30,
        check=True,
        idempotent=True,
    )
    await sb.exec(
        f"chmod +x {launcher}; "
        f"mkdir {lock_dir} 2>/dev/null || exit 0; " # 本次 spawn 的去重 guard
        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 ""
```

tests/test_agent/test_sandbox_exec_and_wait.py

新增语义级回归测试，通过 ShellFakeSandbox 解释 exec_and_wait 实际发出的 shell 字符串来固定去重与清理语义，直接覆盖 main 上失败的 bug 场景，是本 PR 的验证核心。

```
import re
import shlex

# 匹配 exec_and_wait 发出的各类 shell 命令片段
```

```
_POLL_RE = re.compile(r"test -f (\S+) && cat \1")
_SPAWN_RE = re.compile(r"mkdir (\S+) 2>/dev/null \|\| exit 0; (.*)$")
_LAUNCH_RE = re.compile(r"setsid bash (\S+) ")
_TAIL_RE = re.compile(r"tail -c \d+ (\S+)")


class ShellFakeSandbox:
    """解释 exec_and_wait 发出的 shell 命令，作用于内存文件系统。

    run_script 每次"实际启动"被调用一次并返回 (exit_code, output)，
    落盘到 done/out marker 文件，与真实分离命令的写入方式一致。
    """

    sandbox_id = "shell-fake"

    def __init__(self, run_script):
        self.run_script = run_script
        self.files: dict[str, str] = {}
        self.dirs: set[str] = set()
        self.launches = 0
        self.exec_log: list[str] = []

    async def __aenter__(self):
        return self

    async def __aexit__(self, *exc):
        return None

    async def write_file(self, path, content, *, user="root"):
        self.files[path] = content

    async def read_file(self, path, *, user="root"):
        return self.files.get(path, "")

    async def exec(self, cmd, *, user="root", env=None, timeout=120, check=False, idempotent=
    True):
        self.exec_log.append(cmd)

        poll = _POLL_RE.search(cmd)
        if poll:
            path = poll.group(1)
            if path in self.files:
                return 0, self.files[path], ""
            return 1, "", ""

        tail = _TAIL_RE.search(cmd)
        if tail:
            return 0, self.files.get(tail.group(1), "")[-512:], ""
```

```
        spawn = _SPAWN_RE.search(cmd)
        if spawn:
            lock_dir, rest = spawn.groups()
            if lock_dir in self.dirs:
                return 0, "", "" # guard 命中：后续命令不再执行
            self.dirs.add(lock_dir)
            self._run_shell_fragment(rest)
            return 0, "", ""

        # 普通清理命令（rm -rf / rm -f 序列）。
        self._run_shell_fragment(cmd)
        return 0, "", ""

    def _run_shell_fragment(self, fragment):
        """执行 spawn 命令里 guard 后半段或纯清理命令。"""
        for part in fragment.split(";"):
            part = part.strip().rstrip("&").strip()
            if part.startswith(("rm -rf", "rm -f")):
                for token in shlex.split(part)[2:]:
                    self.files.pop(token, None)
                    self.dirs.discard(token)
            launch = _LAUNCH_RE.search(part + " ")
            if launch:
                launcher = launch.group(1)
                assert launcher in self.files, "launcher 必须先于 spawn 写入"
                self.launches += 1
                exit_code, output = self.run_script(self.launches)
                out_file = re.search(r"> (\S+) 2>&1", part).group(1)
                done_file = launcher.replace(".sh", ".done")
                self.files[out_file] = output
                self.files[done_file] = f"{exit_code}\n"
```

评论区精华

该 PR 没有公开 review 评论（comments_count=0, review_comments_count=0），核心论述集中在 PR body 中。PR body 对根因的剖析最有价值：guard 的本来目的是防止同一 spawn RPC 因 sever 响应被 E2BSandbox._rpc_retry 重放而 double-execute；但 lock_dir 从不删除、清理位于 guard 之后，导致 guard 错误地拦住了下一次逻辑调用。修复保留 guard 语义的同时，将清理前置到独立幂等 RPC，并明确文档化调用方不得让同 tag 的两次 exec_and_wait 重叠。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 协议兼容性：Sandbox.exec 新增 idempotent 关键字参数，第三方 sandbox 若未实现该参数，exec_and_wait 调用时仍会 TypeError；本 PR 顺带修复了此不一致，但依赖方

需要同步更新自己的实现。

2. 同 tag 并发约束：修复后清理 RPC 与 spawn RPC 分离，若调用方在同一 tag 上重叠发起两次调用，可能出现清理掉对方启动状态或双 spawn 的竞争；PR 已在调用点注释中明确禁止，但代码层面没有防护。
3. 清理 RPC 失败：清理命令 rm -rf 本身幂等且不易失败，但若 sandbox 文件系统异常导致该 RPC 返回非零，*check=True* 会直接抛异常，可能中断原本可恢复的流程。
4. 测试有效性依赖 shell 语义解析：测试通过正则模拟 shell 行为，覆盖了当前命令布局；若未来命令结构变化但语义不变，测试可能误报或漏报，需要后续维护。 - 影响：影响范围主要是 *agent* 沙箱执行路径：*exec_and_wait* 是 *agent* 示例中构建、测试等长命令的标准执行入口，修复直接改善了 install_npm_cli 等多重试场景的可靠性，避免瞬时失败被静默放大为硬失败。对用户的影响是沙箱命令重试语义变得符合直觉；对系统的影响是减少了一次失败掩盖后续成功机会的静默错误。团队侧新增的语义级测试为后续 sandbox 命令结构调整提供了安全网，CI 增加少量 CPU 测试时间。 - 风险标记：协议签名变更，同 tag 并发约束，shell 语义解析脆弱

关联脉络

- 暂无明显关联 PR