PR #1790 完整报告

radixark/miles

openenv/tbench2: score the shared-server leg natively; retire the adapter compensation

合并时间：2026-07-31 05:31

原文链接：http://prhub.com.cn/radixark/miles/pull/1790

执行摘要

- 一句话：共享服务器腿改原生打分，退役适配补偿
- 推荐动作：值得精读，尤其适合想理解「删除兼容分支时如何保正确性」的工程师。三个看点：(1) 运行时契约守卫的设计——当 preflight 不可行时，用回复自身的契约指纹（info.harness）把静默误打分变成大声失败；(2) 跨仓库边界的演进编排——miles 侧 PR 明确依赖上游 OpenEnv#1012 且给出双向不兼容的切换顺序；(3) review 中 Shi-Dong 抓出的 0.0 vs None 差异，是「删除看似等价代码」时最容易漏掉的行为边界。

功能与动机

PR body 说明这是 #1675 的 follow-up，解决该 PR review 中关于退役 native_evaluate=False 分支的讨论。核心动机有三：(1) 让两条 agent-function leg（共享服务器 / Daytona 沙箱）统一为单一打分协议，消除 adapter 侧针对旧 tbench2_env 部署的补偿机制；(2) 前提是上游先把 docker 模式补齐——OpenEnv#965/#972 只修了 local 模式，docker 模式仍用裸 pytest 从 /task 打分，直接翻转会造成 fidelity regression，因此先等 OpenEnv#1012 合入（04d259ea6）；(3) 面对无法预检源码的远程共享服务器，用运行时契约守卫把「旧部署静默返回看似合理的错误奖励」变成「每个样本大声丢弃」，避免污染训练信号。

实现拆解

1. 退役 adapter 侧补偿机制（openenv_agent_function.py，+75/-162）：删除 _TASK_WORKDIR / _TB2_TESTS_SRC 两个环境变量、_apply_workdir、_CANONICAL_EVAL_CMD、_REWARD_MARKER / _TESTSH_RC_MARKER 及 _parse_reward_marker / _parse_testsh_rc 解析器，以及 testsh_rc 指标管线；模块 docstring中的OPENENV_TASK_WORKDIR/OPENENV_TB2_TESTS_SRC条目随之移除。原因：这些补偿只服务于旧 tbench2_env 部署（裸 pytest 打分、WORKDIR 不在 /app），OpenEnv#1012 后服务端自身具备同款契约，补偿成为必须清理的双协议债务。
2. 统一两条 leg 的 episode 接线：_multi_turn 的关键字参数从三个（run_body / native_evaluate / post_episode）减为两个（run_body / post_episode），native_evaluate 参数整体删除；openenv_daytona_agent_function.py 的 run_episode 不再传 native_evaluate=True。两条腿现在只剩两个真实差异：run_body（环境如何创建：连共享服务器 vs 建 Daytona 沙箱）与 post_episode（共享服务器需要 trial-dir 清理，一次性的沙箱不需要）。

3. 新增运行时契约守卫：新增 _obs_info 辅助函数读取 observation.info；_multi_turn 的 evaluate 处理要求回复携带 info.harness == "tests/test.sh"（当前上游才会发射的标记），否则记警告并以 reward=None 丢弃样本，与 #1675 已有的 error/None-reward 丢弃逻辑叠加。这是对「远程服务器无法做源码 preflight」的替代方案；已知行为变化：不带 tests/test.sh 的任务目录（走服务端 pytest fallback，无 harness 标记）也会被丢弃——这是刻意取舍，因为该路径与旧服务器不可区分，且 89 个官方任务与合成池都带 test.sh。
4. scan_golden.py 采用 no-verdict 语义（+16/-6）：golden 回放的 evaluate 处理从「缺 reward 记 0.0」改为「reward 缺失 / error 非空 / harness 非 canonical 都记为 ERR（reward=None + error 字段）」，避免把服务器端打分失败误记为任务的 0.0 基线；日志尾部捕获条件同步从 reward < 1.0 改为 reward is None or < 1.0。
5. 配套改动：测试侧 _FakeEnv / _FakeResult 改为模拟携带 harness 标记的契约服务器，evaluate 分支直接返回 reward + info；新增 test_old_server_reward_is_not_trusted 用 _OldServerEnv 模拟旧服务器（reward=1.0 但 info 只有 {tests_passed, exit_code}），断言 reward=None 且 episode 本身跑完（turns == 2）。配置侧 OPENENV_MESSAGE_TIMEOUT_S 默认值从 600 提到 1200（evaluate 现在在单次 env 操作内跑完整 verifier 预算，官方任务最高 3600s；超时发生在完整轨迹生成之后，是最昂贵的失败点）。文档侧 README 与 openenv_launch_common.py 的预检报错把安装下限从「#965/#972 合并」改为「≥ #1012 合并（04d259ea6）」并去掉「not upstream main」这类旧表述。

关键文件：

- examples/experimental/openenv/openenv_agent_function.py（模块 环境适配；类别 source；类型 core-logic；符号 _multi_turn, _obs_info, _apply_workdir, _parse_reward_marker）：本 PR 的核心：删除 adapter 侧全套补偿机制（_apply_workdir / _CANONICAL_EVAL_CMD / marker 解析器），新增 _obs_info 与 evaluate 契约守卫，并把 OPENENV_MESSAGE_TIMEOUT_S 默认值从 600s 提到 1200s。净删 162 行，是双协议统一的主战场。
- examples/experimental/openenv/tests/test_openenv_agent_function.py（模块 适配器测试；类别 test；类型 test-coverage；符号 _FakeEnv, _FakeResult, _OldServerEnv, test_shared_leg_dispatch）：测试配套的核心：_FakeEnv / _FakeResult 改为模拟携带 harness 标记的契约服务器，并新增 test_old_server_reward_is_not_trusted 直接验证守卫对旧服务器的拒绝行为（reward=1.0 但无 marker → reward=None）。
- examples/experimental/openenv/scan_golden.py（模块 回放脚本；类别 source；类型 core-logic；符号 scan_one）：golden 回放脚本同步采用 no-verdict 语义：服务器端打分失败或 harness 非 canonical 时记为 ERR（reward=None + error 字段），而不是伪造 0.0 基线，避免把基础设施故障错误归结到任务上。
- examples/experimental/openenv/openenv_daytona_agent_function.py（模块 云沙箱；类别 source；类型 core-logic；符号 run_episode, run）：Daytona 腿的 run_episode 删掉 native_evaluate=True 传参及相关注释，改为与共享腿完全一致的 _multi_turn 接线；两条腿只在 run_body / post_episode 上分化。
- examples/experimental/openenv/openenv_launch_common.py（模块 启动预检；类别 source；类型 core-logic；符号 apply_optional_env_vars）：Daytona 腿的启动预检报错

文案从「安装 pinned checkout，不要上游 main」改为「安装 ≥ #1012 合并（04d259ea6）的 checkout」，消除双协议时代的旧表述。

- examples/experimental/openenv/tests/test_openenv_daytona_agent_function.py（模块 沙箱测试；类别 test；类型 test-coverage；符号 test_daytona_leg_dispatch, test_daytona_leg_eval_error_yields_no_verdict）：Daytona 腿测试同步更新：断言 exec 原样透传 + evaluate 打分，并去掉守护已删除路径的 test.sh 负向断言（那是一条代码已无法产生的路径）。
- examples/experimental/openenv/README.md（模块 文档说明；类别 docs；类型 documentation）：文档把共享服务器的要求写成版本契约（tbench2_env ≥ #1012 合并 04d259ea6），并说明不满足契约时 adapter 会丢弃每个 episode 并告警。

关键符号：_multi_turn, _obs_info, run_episode, test_old_server_reward_is_not_trusted, _apply_workdir（已删除）, _parse_reward_marker（已删除）, _parse_testsh_rc（已删除）

关键源码片段

examples/experimental/openenv/openenv_agent_function.py

本 PR 的核心：删除 adapter 侧全套补偿机制（_apply_workdir / _CANONICAL_EVAL_CMD / marker 解析器），新增 _obs_info 与 evaluate 契约守卫，并把 OPENENV_MESSAGE_TIMEOUT_S 默认值从 600s 提到 1200s。净删 162 行，是双协议统一的主战场。

```
def _obs_info(result: Any) -> dict:
    """读取 StepResult 上 Observation 的 info 字典（缺失时返回空 dict）。

    契约守卫靠它读 evaluate 回复的 info.harness——这是判断服务器是否
    满足 canonical 打分契约的运行时指纹（对远端服务器无法做源码 preflight）。
    """
    obs = getattr(result, "observation", result)
    return getattr(obs, "info", None) or {}


# 运行时契约守卫（_multi_turn 的 evaluate 处理段；逻辑与 scan_golden.py 完全同语义）：
# 旧部署不会报错，只会每个 episode 返回一个看起来合理的（但打错分的）
# reward。所以守卫只认一个证据：evaluate 回复必须携带当前上游
#  （>= OpenEnv#1012，04d259ea6）才会发射的 harness 标记；任何其它情况
# —— 旧服务端 bare-pytest 的 {tests_passed, exit_code} info、服务端打分
# error、reward 缺失 —— 一律按 " 无裁决 " 丢弃样本，而不是记一条假 0.0：
#
# info = _obs_info(res)
# if (getattr(res, "reward", None) is None
# or _obs_field(res, "error")
# or info.get("harness") != "tests/test.sh"):
# logger.warning(
# "evaluate produced no canonical verdict "
# f"(error={_obs_field(res, 'error')!r}, harness={info.get('harness')!r}); "
# "dropping episode"
```

```
# )
# reward = None # 训练包装器据此丢弃样本
# else:
# reward = float(getattr(res, "reward"))
```

examples/experimental/openenv/scan_golden.py

golden 回放脚本同步采用 no-verdict 语义：服务器端打分失败或 harness 非 canonical 时记为 ERR（reward=None + error 字段），而不是伪造 0.0 基线，避免把基础设施故障错误归结到任务上。

```
m = {
    "solve_exit": solve_exit,
    "solve_s": round(solve_s, 1),
    "eval_s": round(time.monotonic() - t, 1),
}
# 与 agent loop 守卫相同的 no-verdict 语义：服务端打分失败
# 或 harness 非 canonical 时记为 ERR，而不是伪造一个 0.0
# 把基础设施故障错记到任务头上。
raw_reward = getattr(res, "reward", None)
eval_error = oaf._obs_field(res, "error")
harness = str(oaf._obs_info(res).get("harness", ""))
if raw_reward is None or eval_error or harness != "tests/test.sh":
    m["reward"] = None
    m["error"] = f"no canonical verdict (error={eval_error!r}, harness={harness!r})"
else:
    m["reward"] = float(raw_reward)
if capture_logs and (m["reward"] is None or m["reward"] < 1.0):
    # evaluate 输出本身携带 test.sh 日志尾部；磁盘上的
    # /logs/verifier 只在 verify 窗口内存在
    m["test_log_tail"] = (oaf._obs_field(res, "output") or "")[-800:]
    res = await env.step(action(action_type="exec", command="tail -c 1200 /tmp/
    solve.log 2>&1"))
    m["solve_log_tail"] = oaf._obs_field(res, "output")
return m
```

评论区精华

两条 review 交锋均来自 Shi-Dong 的逐行核对，质量很高：

1. 0.0 vs None 的语义差异（正确性）：Shi-Dong 对照 OpenEnv#1012 源码（04d259ea6）指出，被删除的 _parse_reward_marker 在 reward.txt 缺失时返回 None（丢弃样本），而上游 evaluate 在 reward.txt 缺失时给 0.0——这意味着删除补偿后，服务端 test.sh 崩溃等 infra 失败会以假 0.0 混入训练信号。nblintao 承认差异并提交 huggingface/OpenEnv#1025 修复上游。
2. 超时默认值 vs 服务端 verifier 预算（性能 / 运维）：Shi-Dong 提示 OpenEnv 默认给 verifier 900 秒，600s 的消息超时默认值会掐断合法 evaluate。nblintao 将 OPENENV_MESSAGE_TIMEOUT_S 默认值提到 1200s 并写入文档（必须超过服务端最长的 verifier budget），同时创建 OpenEnv#1026 建议两侧超时自动协商管理。

- reward.txt 缺失时的语义差异：0.0 vs None (correctness): nblintao 承认差异并提交 huggingface/OpenEnv#1025 修复上游；本 PR 的删除按计划推进，语义差由上游 follow-up 收敛。
- 消息超时默认值与服务端 verifier 预算 (performance): nblintao 将 OPENENV_MESSAGE_TIMEOUT_S 默认值提到 1200s 并写入文档（必须超过服务端最长 verifier budget），另建 OpenEnv#1026 建议两侧超时自动协商管理。

风险与影响

- 风险：
  1. client/server 强耦合的切换窗口：新旧双向不兼容——旧 client 的补偿逻辑在 tests 被 withheld 的新服务器上会失效，新 client 在旧服务器上会丢弃一切样本。PR body 明确要求步骤 2（服务器升级重启）与 3（本 PR 合入）由部署方同步执行，这是最大的运营风险。
1. 残余静默误打分路径：Shi-Dong 指出的 reward.txt 缺失 → 0.0 语义差在 OpenEnv#1025 合入前依然存在：服务端 test.sh 异常退出且 /logs/verifier 无裁决时，上游会给 0.0 而非丢弃，假负样本仍可能进入训练数据。
2. 行为变更波及自定义任务：不带 tests/test.sh 的任务目录现在一律被丢弃（之前走服务端 pytest fallback 还能打分）。官方 89 任务与合成池无影响，自定义任务集需要核对。
3. 超时配置敏感：OPENENV_MESSAGE_TIMEOUT_S 默认 1200s 只保证覆盖服务端默认 900s 预算；声明更大 [verifier].timeout_sec（官方套件最高 3600s）的任务集仍需手动调大，否则会在最昂贵的时点（完整轨迹生成后）失败。
4. 验证缺口：session-server TITO 接线 + GRPO 训练循环未经本 PR 的 GPU 验证（PR 明确 defer 到切换时的 4-GPU smoke），依赖 #1675 的 4×H200 结果覆盖两条腿共享的代码路径。 - 影响：
   1. 部署 / 运维：shared-server 部署必须升级 tbench2_env ≥ 04d259ea6 并重启；升级后旧 client 的补偿逻辑失效，故需与本 PR 同步上线。一个正向副作用：新版服务器把 tests/ 和 solution/ 从 /task 撤走（verifier-asset withholding），同时封掉了旧部署暴露给 agent 的 reward 作弊洞。
5. 系统：agent loop 净减约 200 行，双协议并存的维护复杂度消除；奖励语义与官方 Terminal-Bench-2 对齐（服务端跑 canonical tests/test.sh、真实 WORKDIR、真实 verifier 预算）。
6. 团队：与上游 huggingface/OpenEnv 的契约耦合加深，仓库内多处以「≥ #1012 合并（04d259ea6）」作为安装下限；正在进行的 OpenEnv#1025 / #1026 follow-up 会继续收紧剩余语义差。 - 风险标记：跨仓库协调部署，旧服务端静默误打分风险，行为变更：无 test.sh 任务被丢弃，超时默认依赖服务端预算，GPU 端到端验证缺失

关联脉络

- PR #1675 （标题未在上下文中提供）：本 PR 的 direct 前身：引入 native_evaluate 双协议接线；PR body 明确引用其 review 线程（discussion_r3645321595）作为退役 native_evaluate=False 分支的依据，且 #1675 的 4×H200 验证覆盖了两条腿共享的

agent loop / 训练包装代码，封住了本 PR 的 GPU 验证缺口。