PR #2558 完整报告

radixark/miles

test(ci): move the agentic-env integration tests under tests/fast so CI runs them

合并时间：2026-08-18 03:55

原文链接：http://prhub.com.cn/radixark/miles/pull/2558

执行摘要

- 一句话：15 个示例测试迁入 tests/fast 纳入 CI，并用守卫测试杜绝孤儿测试
- 推荐动作：值得精读，尤其是 tests/ci/test/test_ci_discovery_coverage.py 的三个守卫设计和 git ls-files 替代文件系统遍历的决策——这是“用测试约束测试放置”的治理范式，比一次性迁移更有长期价值。值得关注的决策点：①“无法运行的测试应自我声明 skip 而不是隐形缺席”的原则；②_KNOWN_ORPHANS 设计为只收缩，条目随文件迁移自动过期，避免豁免永久化；③test_the_example_suites_reach_the_cpu_plan 直接调用 collect_tests(discover_ci_files()) 校验真实计划，而不是断言文件存在这种弱检查。对 CI 治理、测试编排感兴趣的读者可重点参考其结构。

功能与动机

关联 Issue #2546 明确提出：“Offline CPU test suites that need no API, no SDK and no GPU should run in CI. Three of them live under examples/ today and are collected by nobody.” PR body 进一步点出代价：“#2545 is what that costs”——一个阻止 16 节点 GB300 任务在 9144 个文件描述符处停摆的回归测试，被加进了没有 CI 任务会打开的文件。PR 确立的规则是：“a test that cannot run somewhere says so itself”——无法运行的测试应当用模块级 skip 自我声明，而不是住在一个缺失不可见的角落。

实现拆解

实现拆解

将 examples/experimental/openenv/tests/ 下 10 个文件、examples/experimental/hud/tests/ 下 4 个文件、examples/experimental/nemo-gym/tests/ 下 1 个文件分别迁入 tests/fast/examples/experimental/{openenv,hud,nemo_gym}/。tests/fast/ 按 ci_register.py 的 _is_implicit_fast_cpu_path 约定自动注册为 CPU CI，无需在 ci_register.py 中显式声明，也无需改动 pyproject.toml 的 testpaths。每个文件的 docstring 同步从“Not collected by the repo-level pytest run”改为“Runs on every PR”，并更新 README 中的手动 pytest 命令。

迁移前每个测试文件自带两条 sys.path.insert(0, str(Path(__file__).resolve().parent.parent))，并依赖 # noqa: E402 绕过“导入不在顶部”的 lint。原因在于 rollout 通过 --custom-agent-function-path 按路径加载示例模块，模块之间用裸名互相 import，测试必须模拟同一方式。改造后每个包内的 __init__.py 定义 EXAMPLE_DIR（从 parents[5] 定位到

checkout 根目录再拼接 examples/experimental/<name>），conftest.py 负责把 EXAMPLE_DIR 插入 sys.path；跨测试 fake 导入（如 from test_openenv_agent_function import _CLASSES）改为包相对导入 from .test_openenv_agent_function import ...。noqa: E402 随样板一起删除。

该文件注册为 stage-a-cpu 自身的一份 CPU 用例，包含三个测试：test_no_test_file_lives_outside_the_tests_tree 遍历“仓库追踪的树外 test 文件”（_KNOWN_ORPHANS 豁免）；test_no_known_orphan_outlives_its_reason 让豁免条目在文件移走后自动报错过期；test_the_example_suites_reach_the_cpu_plan 通过真实调用 collect_tests(discover_ci_files()) 断言 tests/fast/examples/ 下每个条目都进入 CPU 的 stage-a-cpu 计划且未被禁用。_KNOWN_ORPHANS 当前仅含 miles/utils/test_wandb_utils.py（对应 #2557），并带 TODO(#2557) 注释引导删除。

初版守卫用文件系统遍历（从 checkout 根目录 rglob）统计孤儿测试，但 CI 会把 sglang 和 Megatron-LM 克隆进 workspace，导致从 checkout 根遍历会继承其他仓库数千个测试文件。改为 git ls-files -z -- '*test_*.py' 询问“这个仓库追踪了什么”，从语义上把问题限定在仓库自身版本控制的范围内，并用 -z 处理含空格路径。

PR body 给出验证矩阵：pytest tests/fast/examples/experimental/{openenv,hud} → 101 passed、7 skipped、2.1 s；tests/ci/test → 341 passed；run_suite.py --hw cpu --suite stage-a-cpu --list-only 列出全部 15 个文件；并实际栽种一个孤儿文件确认守卫测试变红。nemo-gym 套件在 stage-a-cpu 上 7 个测试全部通过，hud 4 个按设计报告为 skip。

关键文件：

- tests/ci/test/test_ci_discovery_coverage.py（模块 CI 守卫；类别 test；类型 test-coverage；符号 _test_files_outside_the_tests_tree, test_no_test_file_lives_outside_the_tests_tree, test_no_known_orphan_outlives_its_reason, test_the_example_suites_reach_the_cpu_plan）：本 PR 的核心创新：三条守卫规则把“测试文件必须住在 tests/ 下”变成可执行检查，并用 git ls-files 规避 CI 克隆外部仓库导致的文件系统误判。
- tests/fast/examples/experimental/openenv/conftest.py（模块 测试引导；类别 test；类型 test-coverage）：迁移后的统一 sys.path 引导点，替代 10 个测试文件各自携带的 sys.path.insert 样板，是示例测试可移植性的关键设计。
- tests/fast/examples/experimental/openenv/test_openenv_sandbox_common.py（模块 沙箱测试；类别 test；类型 rename-or-move）：openenv 套件中覆盖最广的公共沙箱层测试，直观展示迁移前后的导入形态变化（sys.path.insert + noqa: E402 → 包相对导入 + EXAMPLE_DIR）。
- tests/fast/examples/experimental/openenv/__init__.py（模块 测试定位；类别 test；类型 test-coverage；符号 EXAMPLE_DIR）：定义 EXAMPLE_DIR 作为测试与示例目录之间的唯一定位点，conftest.py 与两个断言示例自身文件的测试都依赖它。
- tests/fast/examples/experimental/nemo_gym/test_nemogym_agent_function.py（模块 适配器测试；类别 test；类型 rename-or-move）：nemo-gym 套件是待定决策的验证样本：依赖 datasets 的测试迁入后首次在 stage-a-cpu 上 7 个用例全部通过，证明 CI 镜像具备运行条件。

- tests/fast/examples/experimental/nemo_gym/conftest.py（模块 测试引导；类别 test；类型 test-coverage）：与 openenv/conftest.py 构成同一引导模式的另一实例，同时也是 review 讨论的焦点（重复代码与 parents[5] 深度约定）。

关键符号：_test_files_outside_the_tests_tree, test_no_test_file_lives_outside_the_tests_tree, test_no_known_orphan_outlives_its_reason, test_the_example_suites_reach_the_cpu_plan

关键源码片段

tests/ci/test/test_ci_discovery_coverage.py

本 PR 的核心创新：三条守卫规则把“测试文件必须住在 tests/ 下”变成可执行检查，并用 git ls-files 规避 CI 克隆外部仓库导致的文件系统误判。

```
"""一个 CI 永远不会打开的测试文件只是文档，不是测试。

pytest 的 testpaths 指向 ./tests，ci_register 在其下扫描四个根目录，所以
放在被测代码旁边的测试文件只有等人手动想起才会运行。仓库里积攒了
15 个这样的文件——openenv 示例就带了 10 个文件、100 个离线测试，它们
只需要 2 秒且不需要 SDK——而 #2545 正是这种安排的代价：针对文件描述符
泄漏（曾让一个 16 节点任务停摆）的回归测试，被加进了一个没有 CI 任务
会收集的文件。

因此这些测试立下的规则是：测试文件必须住在 tests/ 下；暂时无法运行的
测试应当用模块级 skip 自己说明（见 hud 套件的 importorskip）。skip 会
被运行器报告，并在依赖落地那天自动开始工作；而树外的文件无论怎样都
是隐形的。

_KNOWN_ORPHANS 是逃生口，而且它被设计成只会收缩：第三条测试会在
条目所指文件移动后就失败，所以过期的豁免无法残留。
"""

import subprocess
from pathlib import Path, PurePosixPath

import pytest
from tests.ci.ci_register import HWBackend, collect_tests, discover_ci_files, register_cpu_ci

# 自身也是 stage-a-cpu 的一份 CPU 用例，估时 1 秒
register_cpu_ci(est_time=1, suite="stage-a-cpu", labels=[])

REPO_ROOT = Path(__file__).resolve().parents[3]

# 当前被规则豁免的文件。条目存在就是为了被删除；新增一条本身就是一个
# 需要在 review 中捍卫的决定，而不是走过场。
_KNOWN_ORPHANS = {
    # TODO(#2557): 移入 tests/ 并删除本条。
    "miles/utils/test_wandb_utils.py",
}
```

```
def _test_files_outside_the_tests_tree() -> list[str]:
    """问 git 而不是问文件系统：本仓库追踪了哪些树外的测试文件。

    从 checkout 根目录做目录遍历是错的：CI 会把 sglang 和 Megatron-LM
    克隆进工作区，遍历会继承其他仓库的数千个测试文件（本地还会遇到
    venv 或 worktree 里的东西）。
    """
    listing = subprocess.run(
        ["git", "ls-files", "-z", "--", "*test_*.py"],
        cwd=REPO_ROOT,
        capture_output=True,
        text=True,
        check=True,
    ).stdout
    return sorted(
        path
        for path in listing.split("\0")
        if path and not path.startswith("tests/") and PurePosixPath(path).name.startswith("test_")
    )


def test_no_test_file_lives_outside_the_tests_tree():
    if not (REPO_ROOT / ".git").exists():
        pytest.skip("没有 git 元数据可问；规则约束的是仓库追踪的内容")
    offenders = [f for f in _test_files_outside_the_tests_tree() if f not in _KNOWN_ORPHANS]
    assert offenders == [], (
        "这些文件名字像测试却没有运行器收集；请把它们移入 "
        "tests/fast/（CPU）或 tests/fast-gpu/，并在模块级用 skip "
        f"声明缺少的依赖：{offenders}"
    )


def test_no_known_orphan_outlives_its_reason():
    """已经移走的文件留下的豁免条目豁免不了任何东西。"""
    stale = sorted(f for f in _KNOWN_ORPHANS if not (REPO_ROOT / f).is_file())
    assert stale == [], f"这些豁免不再指向存在的文件，请删除：{stale}"


def test_the_example_suites_reach_the_cpu_plan(monkeypatch):
    """移动只有在运行器真的收集它们时才有意义。

    discover_ci_files() 用仓库相对路径做 glob，读的是运行器启动时的 cwd；
    这里把它钉在 checkout 根目录上，再对真实计划做断言。
    """
    monkeypatch.chdir(REPO_ROOT)
    plan = collect_tests(discover_ci_files())

    example_entries = [r for r in plan if r.filename.startswith("tests/fast/examples/")]
    assert example_entries, "tests/fast/examples/ 下没有任何测试进入 CI 计划"
```

```
    for entry in example_entries:
        assert entry.backend == HWBackend.CPU, entry.filename
        assert entry.suite == "stage-a-cpu", entry.filename
        assert entry.disabled is None, entry.filename
```

tests/fast/examples/experimental/openenv/test_openenv_sandbox_common.py

openenv 套件中覆盖最广的公共沙箱层测试，直观展示迁移前后的导入形态变化（sys.path.insert + noqa: E402 → 包相对导入 + EXAMPLE_DIR）。

```
"""共享沙箱层的离线单元测试（无网络、无 GPU）。

现在随每次 PR 运行（stage-a-cpu，遵循 tests/fast/ 约定）；本地运行：

    pytest tests/fast/examples/experimental/openenv -q

覆盖每个后端都继承的公共逻辑，因此只需证明一次而不是每个 provider
各证明一次：后端注册表、episode 分发、sandbox 创建限流的退避与
取消重入等。
"""

import asyncio
import logging
import sys
import threading
from contextlib import asynccontextmanager

# 示例模块由 conftest.py 统一放入 sys.path，不再需要 per-file 的
# sys.path.insert；跨测试的 fake 共享改为包相对导入，noqa: E402 随之删除
import openenv_agent_function as oaf
import openenv_sandbox_common as common
import pytest

from . import EXAMPLE_DIR
from .test_openenv_agent_function import _CLASSES, _FakeEnv, _FakePolicy, _FakeResult

logger = logging.getLogger("test-backend")


def _backend(**overrides) -> common.SandboxBackend:
    """一个 provider 特化部分为桩的 backend。

    每个真实 backend 都是这个对象加上 start hook 与 throttle 分类器，
    所以在这里测试它就等于测试所有 backend。退避被缩小到毫秒级，
    保证重试测试瞬时完成。
    """
    spec = {
        "provider": "Fake",
        "start_sandbox": lambda task_id, tasks_dir: ((lambda: None), "http://sandbox:8000"),
```

```
        "is_throttle": lambda exc: isinstance(exc, _Throttled),
        "logger": logger,
        "backoff_base_s": 0.001,
        "backoff_cap_s": 0.001,
    }
    return common.SandboxBackend(**{**spec, **overrides})


@pytest.mark.parametrize(
    ("name", "expected"),
    [
        ("daytona", "daytona"),
        ("e2b", "e2b"),
        ("agentenv", "e2b"), # agentenv 是 e2b 的别名
        ("  E2B  ", "e2b"),
        ("modal", "modal"),
    ],
)
def test_resolve_backend_normalizes_names_and_aliases(name, expected):
    assert common.resolve_backend(name) == expected


@pytest.mark.parametrize("name", [None, "", "    "])
def test_resolve_backend_refuses_to_pick_for_you(name):
    # 注册表存在的意义就是拒绝猜测：哪个 provider 跑，决定扣谁的配额
    with pytest.raises(ValueError, match="no sandbox backend named"):
        common.resolve_backend(name)
```

tests/fast/examples/experimental/openenv/__init__.py

定义 EXAMPLE_DIR 作为测试与示例目录之间的唯一定位点，conftest.py 与两个断言示例自身文件的测试都依赖它。

```
"""openenv tbench2 示例的测试，示例本体住在测试树之外。

EXAMPLE_DIR 是跨越两者的唯一定位点：conftest 把它放进 sys.path 让
模块按裸名导入；断言示例自身文件的测试（面向操作员的帮助文本、
launcher 传入的 agent-function 目标）从这里读取。
"""

from pathlib import Path

# parents[5] 到达仓库根目录：本文件位于
# tests/fast/examples/experimental/openenv/，上溯 5 级即 checkout 根。
# 一旦包嵌套层级变化，这里会指向错误目录并让相关测试失败——
# 不是静默失效，但确实是一个需要留意的深度约定。
EXAMPLE_DIR = Path(__file__).resolve().parents[5] / "examples" / "experimental" / "openenv"
```

评论区精华

1. conftest.py 重复与 parents[5] 目录深度（claude[bot] → nblintao）：claude[bot] 指出 openenv 与 nemo_gym 两个新增 conftest.py 除 docstring 外字节相同，且两个 __init__.py 都用同样的 Path(__file__).resolve().parents[5] 爬到仓库根，构成“目录深度的静默不变量”，若包嵌套变化会静默指错目录，建议抽取共享 helper（非阻塞）。nblintao 回复 “Not fix. It will not break _silently”——目录深度变化会导致相关测试失败而非静默通过，因此不需要提前抽象。最终未合并 helper，Shi-Dong 以 LGTM 批准。
2. 守卫测试与真实 CI 注册逻辑的一致性（claude[bot]）：claude[bot] 在整体评论中说明它没有照搬 PR 描述，而是对照了 tests/ci/ci_register.py 的真实实现（discover_ci_files、collect_tests、HWBackend、tests/fast/ 隐式 CPU 约定），确认新守卫测试的假设与真实代码一致，这也是其 LGTM 的依据之一。

- conftest.py 重复引导与 parents[5] 目录深度不变量 (design): nblintao 回复 “Not fix. It will not break _silently”，认为目录深度变化会让相关测试失败而不是静默通过，无需提前抽象；最终未合并 helper，Shi-Dong 以 LGTM 批准，该意见作为 known tradeoff 保留。
- 守卫测试假设与 ci_register 真实实现的一致性核验 (testing): 核验结果与真实代码一致，成为 claude[bot] 给出 LGTM 的依据之一；无后续异议。

风险与影响

- 风险：
  1. git ls-files 依赖仓库元数据：_test_files_outside_the_tests_tree 依赖 .git 存在，本地裸 checkout 会走 pytest.skip 分支（有 fallback），CI 环境克隆完整仓库不受影响；但若 CI checkout 配置了 --filter=blob:none 之类的精简模式，git ls-files 仍可用（它读取 index 而非 blob），风险可控。
  2. glob *test_*.py 覆盖范围从严：该模式会捕获任何名字含 test_ 的 .py 文件，包括非 pytest 的辅助脚本（如 some_test_helper.py），可能出现误报，只能通过 _KNOWN_ORPHANS 逐条豁免；这是有意的“从严”设计，但会给后续新增工具类文件带来额外流程成本。
  3. parents[5] 目录深度约定：EXAMPLE_DIR 依赖包嵌套深度固定，若未来把 tests/fast/examples/experimental/ 再加深一层，parents[5] 会指向错误目录。nblintao 认为该失效会表现为测试失败而非静默，但若错误目录恰好存在同名文件，错误信息可能误导排查。低风险。
  4. hud 套件 4 个文件全量 skip：hud 4 个测试在 hud SDK 未安装时模块级 importorskip，CI 中暂时只贡献 skip 记录，价值待依赖落地后兑现，符合既定规则。
  5. 101 个测试首次进入 CI：这些测试此前从未在 CI 环境运行过，存在环境差异导致的偶发失败可能；PR body 记录首次 stage-a-cpu 运行已全部通过（nemo-gym 7 passed、hud 4 skipped），该风险已被实证消除。 - 影响：对 CI 系统：stage-a-cpu 计划新增 15 个文件、约 100+ 用例，总耗时约 2 秒，成本可忽略；CI 从此多了一道“孤儿测试发现守卫”，所有未来 PR 新增测试文件若落在 tests/ 外会直接变红。对测试基础设施：tests/fast/examples/experimental/ 成为示例测试的标准归宿，conftest.py + __init__.py（EXAMPLE_DIR） 成为示例测试的可复用引导模板；_KNOWN_ORPHANS 提供了带自愈机制的临时豁免通道。对开发团队：新增测试的位置成为必须遵守的约定，且守卫测试本身提供了清晰的整改指引（错误信息直接提示移入 tests/fast/ 或

tests/fast-gpu/ 并用模块级 skip 声明依赖）。对 v0.1 milestone：作为测试治理的收口动作，直接服务于发布前质量保障。 - 风险标记：git ls-files 依赖仓库元数据，parents[5] 目录深度约定，hud 套件全量 skip，101 个测试首次进 CI，glob 覆盖范围从严可能误伤

关联脉络

- PR #2545 Reuse one Daytona client per process instead of one per create attempt: 本 PR 的直接触发事件：该 PR 修复的 fd 泄漏回归测试被放进了 CI 永不收集的目录，暴露出"孤儿测试"治理缺口。
- PR #2546 CI: run the offline CPU test suites that live under examples/ (openenv's 100 tests, 1.9s, zero deps): 本 PR closes 的关联 Issue，提供了完整的迁移提案、三套件的依赖判定和成本测算。
- PR #2557 miles/utils/test_wandb_utils.py 的迁移处理：_KNOWN_ORPHANS 中唯一豁免条目的目标处理 PR，TODO(#2557) 注释依赖其落地后自动删除豁免。
- PR #2561 test(ci): file the verifiers tests under the example they test: 同一治理方向的后续 PR：把 verifiers 测试按 example 目录归档进 tests/fast，验证本 PR 确立的"测试住在 tests/ 下"规则被继续执行。