PR #1901 完整报告

radixark/miles

Snapshot the commands and generated configs of every python launch script

合并时间：2026-08-09 18:41

原文链接：http://prhub.com.cn/radixark/miles/pull/1901

执行摘要

- 一句话：为全部 Python 启动脚本建立命令与配置快照测试
- 推荐动作：值得精读。对测试工程师尤其有借鉴价值：环境冻结 + 文件系统冻结 + 命令录制三件套的组合设计，以及 AST 自动发现 entrypoint 的做法；对架构师值得关注 ' 快照测试进 CI 与维护成本 ' 的权衡，以及 #2279 最终将快照移出 CI 的后续走向。

功能与动机

追踪 issue #1837 正在推进大规模启动脚本重构（从 shell 展开模型参数到 Python 预展开、base64 内联文件参数、shell 引号修复等）。启动脚本负责模型下载、Ray 拉起、训练命令拼装，任何回归都会直接导致训练失败且难以定位。PR body 仅标注 'Part of #1837'，但测试文件的 docstring 明确了验收标准：'Every launcher entrypoint must build exactly the recorded shell commands' 与 'An entrypoint that silently does nothing is a broken launcher, not a passing test'。评审者 yueming-yuan 的评论揭示了反面动机：'after this script refactor, I think people will frequently change the .py scripts, so if they don't regenerate the snapshot each time, the CI will fail and make it a bit more complex'——快照既能拦截非预期回归，也会给有意改动增加摩擦，这个权衡贯穿本 PR 的后续命运。

实现拆解

1. 命令录制器（tests/fast/utils/command_recorder.py）：新增 record_commands，通过 monkeypatch 把 command_utils 与 misc 两个模块中的 exec_command、exec_command_all_ray_node 替换为录制函数，命令被追加进列表并返回假输出，使启动脚本的模型转换、Ray 提交等命令只被记录而不真正执行。
2. Python 启动脚本测试基座（tests/fast/launch_scripts/py_harness.py）：提供 freeze_environment（固定 MASTER_ADDR、WANDB_API_KEY、PYTHONPATH 等，并清除 SLURM_JOB_NUM_NODES、RAY_ADDRESS 等宿主机变量）、install_command_recorder（同时把 create_run_id 固定为 FROZEN_RUN_ID，把 save_to_temp_file 替换为伪文件路径，使启动脚本生成的 YAML 等配置也进入快照）、host_filesystem_frozen（monkeypatch Path.exists，只允许 sandbox 与仓库可见，防止机器上已有 checkpoint 导致启动脚本跳过下载步骤）、import_launch_script 与 call_entrypoint（动态 import，并通过 AST 提取非下划线、非 main 的函数作为 entrypoint）。

3. 参数化快照测试（tests/fast/launch_scripts/test_py_launch_scripts.py）：recorded fixture 对全部 Python 启动脚本的所有 entrypoint 参数化；TestEveryLauncherEntrypoint 断言每次录制与快照完全一致，且除显式豁免的 prepare_mxfp8 外每个 entrypoint 必须至少发出命令；TestHostFilesystemIsFrozen 与 TestDiscovery 保护基座自身（发现列表不空转、NPU 脚本确实不可 import、ExecuteTrainConfig 不再受 SLURM 环境串扰）。
4. 快照资产（tests/snapshots/launch_scripts/py/...）：按 脚本相对路径/entrypoint.txt 组织，覆盖 DeepSeek-V3/V3.2/V4、GLM-4.5/5、Kimi、Gemma、FSDP 等脚本；format_recording 将命令按 ### N 分节、参数换行展开，并把伪文件内容以 ### pseudo file N 追加，因此快照不只锁定 argv，还锁定生成的训练配置文件。
5. 基座复用与配套（tests/fast/launch_scripts/sh_harness.py）：将 _sanitize 公开为 sanitize 供 Python 侧复用同一套路径脱敏；快照更新统一由 MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS 环境变量驱动（assert_matches_snapshot 的 compare-or-update）。提交历史还包含了分支上先行的 PYTHONBUFFERED 修复、shell 快照基座等 squash commit，属于本 PR 的前置铺垫。

关键文件：

- tests/fast/launch_scripts/py_harness.py（模块 测试基座；类别 test；类型 test-coverage；符号 Recording, PyLaunchScript, rel, iter_py_launch_scripts）：Python 启动脚本快照测试的核心基座：环境冻结、命令录制、文件系统冻结、AST 入口发现，决定快照的确定性与可复现性。
- tests/fast/launch_scripts/test_py_launch_scripts.py（模块 快照测试；类别 test；类型 test-coverage；符号 _glm_checkpoint, _nemotron_checkpoint, recorded, TestEveryLauncherEntrypoint）：参数化快照测试主体：对全部 Python 启动脚本的所有 entrypoint 断言命令与快照一致，并保证每个 entrypoint 真的发出命令。
- tests/fast/utils/command_recorder.py（模块 命令录制；类别 test；类型 test-coverage；符号 record_commands, fake_exec_command, fake_exec_command_all_ray_node）：命令录制器被 py_harness 与 command_utils 单测共用，是快照测试的底层捕手。
- tests/fast/launch_scripts/sh_harness.py（模块 共享基座；类别 test；类型 test-coverage；符号 sanitize）：将 _sanitize 公开为 sanitize，供 Python 侧复用相同路径脱敏逻辑，是跨 shell/Python 快照复用的衔接点。
- tests/snapshots/launch_scripts/py/scripts/run_deepseek_v4.py/full_train.txt（模块 快照资产；类别 docs；类型 test-fixture）：快照资产代表：展示快照同时覆盖 shell 命令序列与 save_to_temp_file 生成的 TE 精度配置，证明快照不仅锁 argv 还锁配置。

关键符号：record_commands, fake_exec_command, fake_exec_command_all_ray_node, install_command_recorder, host_filesystem_frozen, import_launch_script, call_entrypoint, format_recording, iter_py_launch_scripts, freeze_environment, test_commands_match_snapshot, test_entrypoint_issues_commands, sanitize, assert_matches_snapshot

关键源码片段

tests/fast/launch_scripts/py_harness.py

Python 启动脚本快照测试的核心基座：环境冻结、命令录制、文件系统冻结、AST 入口发现，决定快照的确定性与可复现性。

```
import contextlib
from dataclasses import dataclass
from pathlib import Path
from typing import Iterator

import miles.utils.external_utils.command_utils as command_utils
from tests.fast.launch_scripts.sh_harness import REPO_ROOT
from tests.fast.utils.command_recorder import record_commands

# 固定 run_id 与关键环境变量：快照必须可复现，不能随机器或用户环境漂移
FROZEN_RUN_ID = '260101-000000-000'
_FROZEN_ENV = {
    'MASTER_ADDR': '127.0.0.1',
    'MILES_SCRIPT_ENABLE_RAY_SUBMIT': '1',
    'PYTHONPATH': '/frozen/pythonpath',
    'WANDB_API_KEY': 'frozen-wandb-api-key',
}
# 宿主机变量（Ray 地址、SLURM 分配、NCCL 调优等）会污染录制结果，统一清除
_CLEARED_ENV = (
    'CUDA_VISIBLE_DEVICES',
    'GITHUB_COMMIT_NAME',
    'GLOO_SOCKET_IFNAME',
    'KEEP_MOE_LORA',
    'MILES_SCRIPT_EXTERNAL_RAY',
    'NCCL_DEBUG',
    'NCCL_DEBUG_FILE',
    'NCCL_NVLS_ENABLE',
    'NCCL_SOCKET_IFNAME',
    'NO_PROXY',
    'OPTIMIZER_CPU_OFFLOAD',
    'RAY_ADDRESS',
    'SLURM_JOB_NUM_NODES',
)


@dataclass(frozen=True)
class Recording:
    commands: list[str] # 录制到的全部 shell 命令
    pseudo_files: list[str] # save_to_temp_file 生成的配置内容


def install_command_recorder(monkeypatch) -> Recording:
    """替换命令执行入口为录制器，并把临时文件替换为伪文件，让快照包含生成的配置。"""
    recording = Recording(commands=record_commands(monkeypatch), pseudo_files=[])
```

```
    def fake_save_to_temp_file(text: str, ext: str) -> str:
        recording.pseudo_files.append(text)
        return f'/frozen/pseudo_file_{len(recording.pseudo_files)}.{ext}'

    monkeypatch.setattr(command_utils, 'create_run_id', lambda: FROZEN_RUN_ID)
    monkeypatch.setattr(command_utils, 'save_to_temp_file', fake_save_to_temp_file)
    return recording


@contextmanager
def host_filesystem_frozen(sandbox: Path) -> Iterator[None]:
    """只允许 sandbox 与仓库目录可见，其余路径一律报告不存在。

    启动脚本会因“模型已存在”而跳过下载，若不隔离，快照内容将
    取决于机器上恰好存在的 checkpoint；Python 3.11 下对不可 stat
    的 /root 路径调用 exists() 会抛 PermissionError，这里统一改写为
    报告不存在。仓库保持可见，因为启动脚本合法地从中解析自己的
    模型参数脚本。
    """
    visible_roots = (sandbox, REPO_ROOT)
    real_exists = Path.exists

    def exists(self: Path, **kwargs: object) -> bool:
        if any(self == root or self.is_relative_to(root) for root in visible_roots):
            return real_exists(self, **kwargs)
        return False

    Path.exists = exists
    try:
        yield
    finally:
        Path.exists = real_exists
```

tests/fast/launch_scripts/test_py_launch_scripts.py

参数化快照测试主体：对全部 Python 启动脚本的所有 entrypoint 断言命令与快照一致，并保证每个 entrypoint 真的发出命令。

```
import json
from pathlib import Path

import pytest

from tests.fast.launch_scripts.py_harness import (
    call_entrypoint,
    format_recording,
    freeze_environment,
    import_launch_script,
    install_command_recorder,
    iter_py_launch_scripts,
```

```
)
from tests.fast.launch_scripts.sh_harness import REPO_ROOT, assert_matches_snapshot

_SNAPSHOT_DIR = REPO_ROOT / 'tests' / 'snapshots' / 'launch_scripts' / 'py'

# NPU patch 专属脚本在本 checkout 无法 import，单独登记并断言其确实不可用
_SCRIPTS_IMPORTABLE_ONLY_UNDER_THE_NPU_PATCH = {'scripts/run_qwen3_4b_npu.py'}

# 默认参数在测试 sandbox 里不可用的脚本：先用 override 伪造最小 checkpoint，
# 例如 GLM 需要 config.json 声明 model_type 与层数才能通过模型配置校验
_SCRIPTS_WHOSE_DEFAULTS_ARE_UNSUPPORTED = {
    'scripts/run_glm5_744b_a40b.py': lambda sandbox: _glm_checkpoint(sandbox, 'GLM-5', 78),
    'scripts/run_nemotron_3_nano_4b_fsdp.py': _nemotron_checkpoint,
}

_ENTRYPOINTS_DISABLED_BY_THEIR_OWN_DEFAULTS = {('scripts/run_deepseek_v4.py',
'prepare_mxfp8')}

_SCRIPTS = [s for s in iter_py_launch_scripts() if s.rel not in _SCRIPTS_IMPORTABLE_ONLY_
UNDER_THE_NPU_PATCH]
_CASES = [(script.rel, entrypoint) for script in _SCRIPTS for entrypoint in script.entrypoints]


@pytest.fixture(params=_CASES, ids=[f'{rel}::{entrypoint}' for rel, entrypoint in _CASES])
def recorded(request, monkeypatch, tmp_path):
    rel, entrypoint = request.param
    freeze_environment(monkeypatch) # 先固定环境
    recording = install_command_recorder(monkeypatch) # 再装命令录制器
    module = import_launch_script(REPO_ROOT / rel) # 按脚本路径动态 import
    overrides = _SCRIPTS_WHOSE_DEFAULTS_ARE_UNSUPPORTED.get(rel, lambda sandbox: {})
    (tmp_path)
    call_entrypoint(module, entrypoint, overrides, sandbox=tmp_path) # 真正执行入口逻辑
    return rel, entrypoint, recording, tmp_path


class TestEveryLauncherEntrypoint:
    def test_commands_match_snapshot(self, recorded):
        """每个启动器入口都必须精确重现快照中的命令与生成的配置。"""
        rel, entrypoint, recording, sandbox = recorded
        snapshot = _SNAPSHOT_DIR / rel / f'{entrypoint}.txt'
        assert_matches_snapshot(snapshot, format_recording(recording, sandbox=sandbox),
        f'{rel}::{entrypoint}')

    def test_entrypoint_issues_commands(self, recorded):
        """静默什么都不做的入口是坏启动器，而不是一个通过的测试。"""
        rel, entrypoint, recording, _ = recorded
        if (rel, entrypoint) in _ENTRYPOINTS_DISABLED_BY_THEIR_OWN_DEFAULTS:
            assert not recording.commands
        else:
```

```
assert recording.commands
```

评论区精华

唯一实质讨论来自关联 issue 的评论（与 #1899 相同的顾虑）：yueming-yuan 认为脚本重构后开发者会频繁改动 .py 脚本，若不每次重新生成快照，CI 就会失败，使开发流程更复杂；多数脚本修改是语义上有意的且只有几行，建议把该测试移到 manual，只在大型改动 / 重构时运行。作者 fzyzcjy 以 '(discussed offline)' 收尾，本 PR 未改变将快照测试保留在 CI 的决定；后续 #2279 最终将这些启动脚本快照测试移出 CI，改为手动执行，印证了该顾虑。

- 快照测试是否应移出 CI 以降低维护成本 (testing): 作者 fzyzcjy 仅回复 '(discussed offline)'。本 PR 未调整，快照测试保留在 CI；后续 #2279 最终将这些启动脚本快照测试移出 CI 改为手动执行，印证了该顾虑。

风险与影响

- 风险：
  - 快照脆弱性：host_filesystem_frozen 依赖全局替换 Path.exists；若启动脚本改用 os.path.exists、Path.is_file 等 API 检查文件，冻结会静默失效，快照重新开始跟随机器状态。测试内部用 TestHostFilesystemIsFrozen 自我验证，但并未覆盖其他文件 API。
  - 维护摩擦：约 60 个快照覆盖全部入口，任何有意的脚本改动都必须用 MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS=1 重新生成快照，漏一步 CI 即红；这也是评审中最主要的反对意见，后续 #2279 移出 CI 后该摩擦转嫁给手动流程。
  - 快照更新机制误用：assert_matches_snapshot 在设置了更新环境变量时静默覆盖快照，误用会掩盖真实回归。
  - SLURM 时序：commit 7fd6269 修复 ExecuteTrainConfig.num_nodes 在 import 时读取 SLURM_JOB_NUM_NODES 的问题（改用 default_factory，并在测试中断言 num_nodes == 1），未来若有人重新在 import 时读环境变量，测试会首先暴露。
  - 影响面：全部为测试侧变更，产品训练路径零改动，无运行时回归风险；CI fast 套件用例数显著增加，但均为进程内录制、无外部依赖。
- 影响：
  - 开发者：修改 scripts/run_*.py 后必须用 MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS=1 重新生成快照；新增脚本会被自动发现并强制要求快照。
  - CI：fast 测试集新增约 60 个快照用例，均为轻量进程内录制；后续因维护成本改手动执行。
  - 团队：为后续 #1905-#1911 启动命令重构提供安全网，任何命令拼装回归都会被快速定位到具体 entrypoint。
  - 风险标记：快照维护摩擦高，依赖宿主机状态隔离，CI 用例量显著增加，快照更新机制可误用，仅测试侧变更

关联脉络

- PR #1899 Snapshot the external commands of every shell launch script: 本 PR 的直接先导，建立了 shell 启动脚本快照；py_harness 复用了其 sanitize、assert_matches_snapshot 与统一快照目录。
- PR #1902 Cover the public surface of command_utils with unit tests: 提交信息明确 'Share the command recorder with the command_utils tests'，两者共享 command_recorder 录制器。
- PR #1909 Expand the model args in python before building the command: 本 PR 快照网保护的后续重构：命令构建方式变化会被快照测试捕获并强制同步更新。
- PR #1911 Quote the model args miles inlines into the launch command: 同样修改启动命令并同步更新快照，是快照网生效的直接案例。
- PR #2279 Run the launch script snapshot tests by hand instead of in CI: 最终回应了本 PR 评审中关于 CI 频繁失败的维护成本担忧，将启动脚本快照测试移出 CI。