PR #1899 完整报告

radixark/miles

Snapshot the external commands of every shell launch script

合并时间：2026-08-09 18:33

原文链接：http://prhub.com.cn/radixark/miles/pull/1899

执行摘要

- 一句话：为全部 shell 启动脚本建立外部命令快照测试
- 推荐动作：值得精读，尤其是 sh_harness.py 的 shim 与冻结环境设计，以及 TestDiscovery 的发现 - 快照双射校验，这些模式可以复用到其他 CLI/ 脚本类仓库。对于维护者，建议关注快照更新机制的滥用风险，并参考 PR 2279 的做法合理规划快照测试的 CI 执行频率。

功能与动机

该 PR 属于 #1837 追踪的启动脚本重构与增强系列，目标是给每个 shell 启动脚本的外部命令做快照，防止后续对启动命令格式、参数顺序或 ray job submit 调用链的改动在不知不觉中破坏已有脚本。issue 评论中 yueming-yuan 也提出了对 CI 复杂度与 .sh 脚本长期维护的疑虑，促使团队在自动化防护与维护成本之间做权衡。

实现拆解

实现分为 5 个步骤：

1. 扩展测试底座 tests/fast/launch_scripts/sh_harness.py：新增 iter_launch_scripts() 按 ray job submit 关键字发现脚本；run_launch_script() 新增 args 透传；扩充 shim 命令白名单（awk、ps、scp、ssh 等），并让 python/python3 的 shim 拦截 cluster_resources 查询与 import 探测，其余参数透传给真实 Python，避免资源查询依赖真实集群。同时冻结日期、进程列表等输出，保证快照可复现。
2. 新增主测试文件 tests/fast/launch_scripts/test_sh_launch_scripts.py：用 LaunchScriptCase 描述需要显式参数或环境变量的脚本（如 p2p 参数、BASE_FOLDER、OUTPUT_DIR 等）；recorded fixture 参数化执行每个脚本；TestEveryLaunchScript 校验调用序列与快照一致，且恰好有一次 ray job submit；TestDiscovery 校验发现集合与快照集合双向一致，防止脚本悄悄脱离覆盖。
3. 批量生成快照：将每条外部命令的 argv 与返回码写入 tests/snapshots/launch_scripts/sh/ 下与脚本同路径的 .txt 文件，作为后续比较基准；快照可通过 MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS 环境变量重新生成。
4. 合入系列前置修复：分支合并中带入了 PYTHONBUFFERED 拼写修正、缺失行连接导致的路径不可解析、硬编码 checkout 位置改为自动推导等变更，这些修复是快照能够稳定落盘的前提。

5. 配套测试：除主测试外，harness 自身行为（fork 顺序、shim 输出、冻结环境拒绝解冻）也有针对性测试覆盖，确保底座本身可信。

关键文件：

- tests/fast/launch_scripts/test_sh_launch_scripts.py（模块 启动脚本；类别 test；类型 test-coverage；符号 LaunchScriptCase, recorded, TestEveryLaunchScript, test_invocations_match_snapshot）：新增的 shell 启动脚本快照主测试，是本次变更的核心。遍历发现的全部启动脚本，校验外部命令调用序列与快照的一致性，并确保恰好一次 ray job submit。
- tests/fast/launch_scripts/sh_harness.py（模块 测试底座；类别 test；类型 test-coverage；符号 iter_launch_scripts, assert_matches_snapshot）：沙箱测试底座，定义了脚本发现、命令 shim、冻结环境、快照比较 / 再生成机制，所有快照测试都依赖它。
- tests/snapshots/launch_scripts/sh/examples/on_policy_distillation/run-qwen3-8B-opd.sh.txt（模块 快照数据；类别 docs；类型 documentation）：代表性快照，展示 harness 记录的外部命令序列格式（sglang 启动、curl 健康检查、ray start、ray job submit、清理命令）；60 余份同类快照是测试对比的基准。

关键符号：iter_launch_scripts, assert_matches_snapshot, run_launch_script, format_invocations, recorded, test_invocations_match_snapshot, test_submits_exactly_one_ray_job, test_every_discovered_script_has_a_snapshot_and_vice_versa

关键源码片段

tests/fast/launch_scripts/test_sh_launch_scripts.py

新增的 shell 启动脚本快照主测试，是本次变更的核心。遍历发现的全部启动脚本，校验外部命令调用序列与快照的一致性，并确保恰好一次 ray job submit。

```
from dataclasses import dataclass, field

import pytest
from tests.fast.launch_scripts.sh_harness import (
    REPO_ROOT,
    assert_matches_snapshot,
    format_invocations,
    iter_launch_scripts,
    run_launch_script,
)

_SNAPSHOT_DIR = REPO_ROOT / "tests" / "snapshots" / "launch_scripts" / "sh"

# 快照基准路径与脚本相对仓库根路径保持一致


@dataclass(frozen=True)
class LaunchScriptCase:
    # 部分脚本必须有显式参数或环境变量才肯继续执行，
```

```
    # 测试需要为它们提供最小可信输入。
    args: tuple[str, ...] = ()
    env: dict[str, str] = field(default_factory=dict)


_CHECKPOINT_DIR = "/frozen/checkpoints"
_HEAD_NODE_IP = "10.0.0.1"

# 需要显式输入才能跑完的脚本清单，按相对仓库根路径索引；
# `{workdir}` 会被替换成沙箱临时目录，保证快照可复现。
_SCRIPTS_REFUSING_TO_RUN_WITHOUT_EXPLICIT_INPUTS: dict[str, LaunchScriptCase] = {
    "examples/lora/run-qwen2.5-3B-megatron-lora-disaggregated-multi-node.sh":
    LaunchScriptCase(args=("p2p", "0")),
    "examples/infra_features/p2p_weight_transfer/run-glm5-disagg-profile.sh":
    LaunchScriptCase(
        args=("GLM-5", "p2p", "0", _HEAD_NODE_IP), env={"MILES_LOG_DIR": "{workdir}"}
    ),
    "scripts/run-qwen3-235B-A22B.sh": LaunchScriptCase(env={"BASE_FOLDER": _CHECKPOINT_
    DIR}),
    "scripts/run-qwen3.6-27B.sh": LaunchScriptCase(env={"OUTPUT_DIR": _CHECKPOINT_DIR}),
}

# 发现所有会执行 `ray job submit` 的 shell 启动脚本
_SCRIPTS = [script.relative_to(REPO_ROOT).as_posix() for script in iter_launch_scripts()]


@pytest.fixture(params=_SCRIPTS, scope="module")
def recorded(request, tmp_path_factory):
    rel = request.param
    case = _SCRIPTS_REFUSING_TO_RUN_WITHOUT_EXPLICIT_INPUTS.get(rel,
    LaunchScriptCase())
    tmp_path = tmp_path_factory.mktemp("launch_script")
    workdir = tmp_path / "workdir"
    run = run_launch_script(
        REPO_ROOT / rel,
        sandbox=tmp_path,
        args=case.args,
        extra_env={key: value.format(workdir=workdir) for key, value in case.env.items()},
    )
    return rel, run


class TestEveryLaunchScript:
    def test_invocations_match_snapshot(self, recorded):
        # 把脚本真实发出的外部命令序列和快照逐条对比，
        # 任何改动（哪怕只是参数顺序）都会在这里暴露。
        rel, run = recorded
        snapshot = _SNAPSHOT_DIR / f"{rel}.txt"
        actual = f"# returncode: {run.returncode}
```

```
{format_invocations(run.invocations)}"
        assert_matches_snapshot(snapshot, actual, rel)

    def test_submits_exactly_one_ray_job(self, recorded):
        # 启动脚本必须恰好提交一次 ray job，否则说明调用链已断。
        _, run = recorded
        assert run.returncode == 0
        assert len(run.ray_job_submit_argv()) > 10


class TestDiscovery:
    def test_every_discovered_script_has_a_snapshot_and_vice_versa(self):
        # 发现集合与快照集合必须双向一致，
        # 避免脚本悄悄脱离覆盖范围也没有任何信号。
        discovered = {f"{rel}.txt" for rel in _SCRIPTS}
        recorded = {path.relative_to(_SNAPSHOT_DIR).as_posix() for path in _SNAPSHOT_DIR.
        rglob("*.txt")}
        assert discovered == recorded
        assert len(discovered) > 60
```

tests/fast/launch_scripts/sh_harness.py

沙箱测试底座，定义了脚本发现、命令 shim、冻结环境、快照比较 / 再生成机制，所有快照测试都依赖它。

```
import os
import sys
from pathlib import Path

_ARG_SEPARATOR = "\x1f"
_RECORD_SEPARATOR = "\x1e"

_SYSTEM_PATH = "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"

# 冻结环境：确保快照不依赖开发机或 CI 宿主机的注入变量
_FROZEN_ENV = {
    "HOME": "/root",
    "LANG": "C",
    "LC_ALL": "C",
    "TERM": "dumb",
    "MASTER_ADDR": "127.0.0.1",
    "NODE_RANK": "0",
    "WANDB_KEY": "frozen-wandb-key",
    "WANDB_API_KEY": "frozen-wandb-api-key",
}

# 需要被替换成假命令的可执行文件白名单；
# 新增真实命令前先确认是否会打破沙箱隔离。
_SHIMMED_COMMANDS = (
    "apt", "apt-get", "awk", "curl", "date", "docker", "git", "hf", "ip", "mkdir",
```

```
    "nc", "nvidia-smi", "pip", "pip3", "pkill", "ps", "python", "python3", "ray",
    "rm", "rsync", "scp", "sleep", "ssh", "torchrun", "wget",
)

# 让“等待 N 个 GPU 加入”的轮询立即退出，而不是死等
_GPU_COUNT_LARGER_THAN_ANY_WAIT_LOOP_EXPECTS = "1000000"

# 冻结 `ps` 输出，避免录制到宿主机的真实进程列表
_FROZEN_RAY_DASHBOARD_PROCESS = "root 1 0.0 0.0 ray dashboard --node-ip-address=10.0.
0.1 --dashboard-port=8265"

_SHIM_STDOUT = {
    "date": "20260101_000000",
    "ps": _FROZEN_RAY_DASHBOARD_PROCESS,
}

# python shim：拦截集群资源查询与 import 探测，其余透传真实 Python
_PYTHON_SHIM_BODY = """case "${1:-}" in
-c)
    case "$2" in
    *cluster_resources*) printf '%s\\n' 'REPLACE_GPU_COUNT' ;;
    *import*) ;;
    *) "$MILES_SH_HARNESS_REAL_PYTHON" "$@" ;;
    esac
    ;;
esac
""".replace("REPLACE_GPU_COUNT", _GPU_COUNT_LARGER_THAN_ANY_WAIT_LOOP_EXPECTS)

_SHIM_BODY = {"python": _PYTHON_SHIM_BODY, "python3": _PYTHON_SHIM_BODY}



def iter_launch_scripts() -> list[Path]:
    # 用“包含 `ray job submit`”作为发现条件，避免把普通工具脚本纳入
    roots = [REPO_ROOT / "scripts", REPO_ROOT / "examples"]
    return sorted(
        path for root in roots for path in root.rglob("*.sh") if "ray job submit" in path.read_
        text(errors="replace")
    )



SNAPSHOT_UPDATE_ENV_VAR = "MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS"



def assert_matches_snapshot(snapshot: Path, actual: str, subject: str) -> None:
    # 设置更新环境变量时直接写回快照，用于批量再生成；
    # 平时则严格相等对比，任何漂移都视为失败。
    if os.environ.get(SNAPSHOT_UPDATE_ENV_VAR):
        snapshot.parent.mkdir(parents=True, exist_ok=True)
        snapshot.write_text(actual)
```

```
        return
    assert snapshot.exists(), f"missing snapshot for {subject}; regenerate with {SNAPSHOT_
    UPDATE_ENV_VAR}=1"
    assert actual == snapshot.read_text()
```

评论区精华

issue 评论中 yueming-yuan 提出两个疑虑：一是担心保留此快照测试会让未来 CI 更复杂，询问合入后是否移除；二是建议把所有 .sh 脚本转换成 .py。作者未在 PR 内直接回应，快照测试随 PR 合并；后续 PR 2279 将快照测试改为手动执行、PR 1910 开始把模型配置 shell 脚本改写为 Python，两条疑虑分别得到回应。

- 快照测试在未来 CI 中的去留 (design): 作者未在 PR 内回应，快照测试保留；后续 PR 2279 将其改为手动执行，回应了该担忧。
- 将 .sh 脚本转换为 .py (design): 本 PR 未做转换；后续系列（PR 1910 等）将模型配置 shell 脚本重写为 Python，方向部分吻合。

风险与影响

- 风险：
  1. 快照严格相等比对对顺序敏感：任何命令行顺序调整都会触发失败，这是设计目标，但也意味着无关联的重排需要人工确认。
  2. shim 命令白名单有限：tests/fast/launch_scripts/sh_harness.py 中未列出的外部命令会在沙箱中真实执行，可能出现网络访问、挂起或污染快照；当前通过 120 秒超时与进程组清理兜底。
  3. 快照更新模式（MILES_UPDATE_LAUNCH_SCRIPT_SNAPSHOTS）会把当前输出直接写回快照，误用时可能掩盖真实回归，需配合 review 人工确认。
  4. CI 时长影响：60+ 脚本逐一沙箱执行，超时上限 120 秒 / 脚本，后续 PR 2279 将该测试移出手工 CI 以缓解压力。 - 影响：对用户无直接行为影响；对团队是在 scripts/、examples/ 启动脚本上新增硬性回归门槛，任何改动启动命令的 PR 都需要同步更新快照。测试底座被后续多个 PR（如 1901、1906、1908）复用，成为启动脚本系列重构的公共基础设施，长期降低启动命令回归的成本。 - 风险标记：快照严格比对易误报，沙箱 shim 覆盖有限，快照更新模式可能掩盖回归，CI 时长增长

关联脉络

- PR #1896 Add a shell launch script test harness for future protection: 本 PR 依赖的沙箱 harness 由该 PR 引入，其提交也包含在本分支历史中。
- PR #1900 Read the slurm allocation when the train config is built: harness 暴露了 SLURM 环境在 import 时被固化的时序问题，属于同一 harness 覆盖下的修复。
- PR #1901 Snapshot the commands and generated configs of every python launch script: 把同一套快照思路扩展到 Python 启动脚本，共享 py_harness 与快照目录。
- PR #1906 Snapshot the launchers that build their own command line: 对自建命令行的启动器补充同类快照，完善启动脚本回归防线。

- PR #2279 Run the launch script snapshot tests by hand instead of in CI: 后续将启动脚本快照测试移出 CI，回应 yueming-yuan 对 CI 复杂度的担忧。