PR #1895 完整报告

radixark/miles

Fix typo environment variable and unbuffer python outputs

合并时间：2026-08-09 18:19

原文链接：http://prhub.com.cn/radixark/miles/pull/1895

执行摘要

- 一句话：修正 PYTHONBUFFERED 拼写错误并开启 Ray 无缓冲输出
- 推荐动作：值得快速精读，重点是两点设计决策：一是“提交客户端 export + runtime env 双通道”的完整无缓冲方案，揭示了 Ray 作业环境变量的传播边界；二是测试文件用 git ls-files 自动发现用例并配合守卫测试防空转的思路，在脚本类仓库中非常有借鉴价值。

功能与动机

PR body 标注为 Part of #1837。核心动机有二：一是 “PYTHONBUFFERED” 是错误拼写，Python 解释器只识别 “PYTHONUNBUFFERED”，错误写法会被静默忽略，Ray 因此一直缓冲 worker 日志，训练过程无法实时观测；二是旧实现只在提交客户端侧 export 变量，没有把它写入 “--runtime-env-json”，Ray worker 进程读到的是 runtime env，所以真正干活的 worker 仍然处于缓冲状态。新测试的 docstring 直接点明问题：“Ray buffers worker stdout unless PYTHONUNBUFFERED rides along with the job it submits”。

实现拆解

按以下 4 步完成（对应 squash 后的 4 个原子提交）：

1. 修正命令构建核心逻辑：miles/utils/external_utils/command_utils.py 的 execute_train 中，将 ray start 与 ray job submit 两处的 export PYTHONBUFFERED=16 改为 export PYTHONUNBUFFERED=1，并在 runtime_env_vars 字典里新增 "PYTHONUNBUFFERED": "1"。这样提交客户端（export 生效）与 Ray worker（runtime env 生效）两条链路都拿到正确变量，且取值从无意义的 16 修正为标准的 1。
2. 全仓批量替换启动脚本：覆盖 examples、scripts、tools 等目录下共 75 个文件，把所有 export PYTHONBUFFERED=16 统一替换为 export PYTHONUNBUFFERED=1。对自建 runtime env 的 Python 启动器（如 examples/experimental/formal_math/single_round/run_minimal.py、examples/infra_features/p2p_weight_transfer/run.py、tools/convert_torch_dist_to_hf_ray.py），除修正拼写外还额外在 runtime env JSON 中补入 PYTHONUNBUFFERED: "1"，保证 worker 侧同样无缓冲。
3. 同步 NPU 部署补丁：docker/npu_patch/miles.patch 中对应的两处 PYTHONBUFFERED=16 一并修正，避免 NPU 镜像构建出的代码与主仓行为不一致。
4. 新增防回归测试：新增 tests/fast/test_ray_launcher_unbuffering.py，通过 git ls-files 扫描 examples/scripts/tools/miles/utils/external_utils 下所有受版本控制的 .py/.sh 文件

，用 runtime env 标记识别 Ray 启动器，参数化断言每个启动器都包含 PYTHONUNBUFFERED 且不含错误拼写 PYTHONBUFFERED，并设置守卫测试（用例数 > 50）防止发现逻辑失效导致测试空转。tests/fast/utils/test_command_utils.py 新增两个单测，分别验证提交客户端侧 export 数量与 runtime env JSON 中 worker 侧变量均正确。

关键文件：

- tests/fast/test_ray_launcher_unbuffering.py（模块 启动器检查；类别 test；类型 test-coverage；符号 tracked_files, ray_launchers, test_the_repo_has_ray_launchers_to_check, test_every_ray_launcher_unbuffers_python）：新增的核心防回归测试：自动发现全仓 Ray 启动器并参数化校验 PYTHONUNBUFFERED 拼写，守卫测试防止发现逻辑失效，是本次改动质量的保障
- miles/utils/external_utils/command_utils.py（模块 命令工具；类别 source；类型 core-logic；符号 execute_train）：核心逻辑修改点：execute_train 同时修正了提交客户端与 Ray worker 两侧的变量，影响所有经它提交的训练作业
- tests/fast/utils/test_command_utils.py（模块 命令工具；类别 test；类型 test-coverage；符号 test_execute_train_exports_unbuffered_python_to_ray, test_execute_train_unbuffers_the_ray_workers_too）：为 execute_train 新增两个单测，分别锁定提交客户端 export 与 Ray worker runtime env 两侧行为
- tools/convert_torch_dist_to_hf_ray.py（模块 转换工具；类别 source；类型 core-logic；符号 make_conversion_actor）：自建 Ray runtime env 的转换工具，需额外为 worker actor 注入 PYTHONUNBUFFERED，是“双通道”修复模式的代表
- docker/npu_patch/miles.patch（模块 部署补丁；类别 infra；类型 infrastructure）：NPU docker 补丁同步修正两处拼写，保证 NPU 部署与主仓行为一致

关键符号：execute_train, tracked_files, ray_launchers, test_the_repo_has_ray_launchers_to_check, test_every_ray_launcher_unbuffers_python, test_no_ray_launcher_spells_the_variable_wrong, test_execute_train_exports_unbuffered_python_to_ray, test_execute_train_unbuffers_the_ray_workers_too, make_conversion_actor

关键源码片段

tests/fast/test_ray_launcher_unbuffering.py

新增的核心防回归测试：自动发现全仓 Ray 启动器并参数化校验 PYTHONUNBUFFERED 拼写，守卫测试防止发现逻辑失效，是本次改动质量的保障

```
import subprocess
from pathlib import Path

import pytest

REPO_ROOT = Path(__file__).resolve().parents[2]
# 扫描范围：所有可能向 Ray 提交作业的启动器所在目录
LAUNCHER_DIRS = ("examples", "scripts", "tools", "miles/utils/external_utils")
```

```
# 脚本里只要出现 runtime env 相关写法，就认为它是 Ray 启动器
RAY_RUNTIME_ENV_MARKERS = ("runtime-env-json", "runtime_env=", "runtime_env_json")


def tracked_files() -> list[Path]:
    # 用 git ls-files 获取受版本控制的文件，避免本地临时脚本干扰检查结果
    listing = subprocess.run(
        ["git", "-C", str(REPO_ROOT), "ls-files", "-z", *LAUNCHER_DIRS],
        capture_output=True,
        text=True,
        check=True,
    )
    return [REPO_ROOT / name for name in listing.stdout.split("\0") if name.endswith((".py", ".sh")
    )]


def ray_launchers() -> list[Path]:
    # 参数化测试的用例来源：按 runtime env 标记筛出真正的 Ray 启动器
    return [path for path in tracked_files() if any(marker in path.read_text() for marker in RAY_
    RUNTIME_ENV_MARKERS)]


def test_the_repo_has_ray_launchers_to_check() -> None:
    """发现逻辑一旦失效会让其他测试静默通过，守卫测试保证用例数量足够多。"""
    assert len(ray_launchers()) > 50


@pytest.mark.parametrize("launcher", ray_launchers(), ids=lambda path: str(path.relative_
to(REPO_ROOT)))
def test_every_ray_launcher_unbuffers_python(launcher: Path) -> None:
    """Ray 默认缓冲 worker 的 stdout，提交作业时必须带上 PYTHONUNBUFFERED
    才能实时看到日志。"""
    assert "PYTHONUNBUFFERED" in launcher.read_text()


@pytest.mark.parametrize("launcher", ray_launchers(), ids=lambda path: str(path.relative_
to(REPO_ROOT)))
def test_no_ray_launcher_spells_the_variable_wrong(launcher: Path) -> None:
    """PYTHONBUFFERED 拼写错误会被 Python 静默忽略，等于没有关闭缓冲。"""
    assert "PYTHONBUFFERED" not in launcher.read_text().replace("PYTHONUNBUFFERED", "")
```

miles/utils/external_utils/command_utils.py

核心逻辑修改点：*execute*_train 同时修正了提交客户端与 Ray worker 两侧的变量，影响所有经它提交的训练作业

```
    external_ray = get_bool_env_var("MILES_SCRIPT_EXTERNAL_RAY")
    master_addr = os.environ.get("MASTER_ADDR", "127.0.0.1")
    train_backend_fsdp = "--train-backend fsdp" in train_args
    assert train_backend_fsdp == (megatron_model_type is None)
```

```
if not external_ray:
    # 注意必须是 PYTHONUNBUFFERED，旧的 PYTHONBUFFERED 是错误拼写，
    # Python 不读取它，Ray 依然会缓冲 stdout/stderr，日志无法实时观测
    exec_command(
        # will prevent ray from buffering stdout/stderr
        f"export PYTHONUNBUFFERED=1 && "
        f"ray start --head --node-ip-address {master_addr} --num-gpus {num_gpus_per_node} --
        disable-usage-stats"
    )

if (f := before_ray_job_submit) is not None:
    f()

runtime_env_vars = {
    # export 只影响提交客户端进程，Ray worker 读的是 runtime env，
    # 因此这里必须再注入一次，保证 worker 侧日志也是无缓冲输出
    "PYTHONUNBUFFERED": "1",
    # FSDP 后端设置 CUDA_DEVICE_MAX_CONNECTIONS 会干扰计算通信重叠，按后端区分
    **({} if train_backend_fsdp else {"CUDA_DEVICE_MAX_CONNECTIONS": "1"}),
    "NCCL_NVLS_ENABLE": os.environ.get("NCCL_NVLS_ENABLE", str(int(check_has_nvlink()))
    ),
    **{
        k: os.environ[k]
        for k in ("NCCL_SOCKET_IFNAME", "GLOO_SOCKET_IFNAME", "NCCL_DEBUG", "NCCL_
        DEBUG_FILE")
        if k in os.environ
    },
    "no_proxy": f"127.0.0.1,{master_addr}",
    # 多节点下 megatron / torch distributed 需要这个变量
    "MASTER_ADDR": master_addr,
    **extra_env_vars,
    **_parse_extra_env_vars(config.extra_env_vars),
}
runtime_env_vars["PYTHONPATH"] = _pythonpath_with_sources(megatron_path, runtime_env_
vars.get("PYTHONPATH"))
runtime_env_json = json.dumps({"env_vars": runtime_env_vars})

if get_bool_env_var("MILES_SCRIPT_ENABLE_RAY_SUBMIT", "1"):
    cmd_megatron_model_source = (
        f'source "{repo_base_dir}/scripts/models/{megatron_model_type}.sh" && '
        if megatron_model_type is not None
        else ""
    )
    exec_command(
        f"export no_proxy=127.0.0.1 && export PYTHONUNBUFFERED=1 && "
        f"{cmd_megatron_model_source}"
        f"""ray job submit {'' if 'RAY_ADDRESS' in os.environ else '--address="http://127.0.0.1:
        8265" '}"""
        f"--runtime-env-json={shlex.quote(runtime_env_json)} "
```

```
        f"-- python3 {train_script} "
        f"{'${MODEL_ARGS[@]}' if megatron_model_type is not None else ''} "
        f"{train_args}"
    )
```

评论区精华

该 PR 没有产生任何 inline review 评论（review_comments_count = 0），reviewer yueming-yuan 直接给出了 APPROVED。从合入方式看，改动属于低争议的机械替换；唯一值得注意的外部噪音是 gemini-code-assist bot 在关联 Issue 下的通知，声明其代码审查服务已停止，与本次变更内容无关。

- 无 review 评论，单次直接批准 (other): 直接合入 main

风险与影响

- 风险：主要风险有三点：

1. 机械替换覆盖面大：75 个文件属于批量替换，存在个别脚本同时出现两个变量名、或因大小写变体未被覆盖的边际情况；新测试只对“含 runtime env 标记”的 Ray 启动器做断言，不含该标记的普通脚本不在检查范围内。
2. 测试盲区：tests/fast/test_ray_launcher_unbuffering.py 基于 git ls-files 扫描，未跟踪或被 gitignore 的本地脚本不受保护；若后续新增启动器不带任何 runtime env 标记，也不会被抓取（但守卫测试在一定程度上缓解了发现逻辑失效的问题）。
3. 行为面变更：runtime env 新增 PYTHONUNBUFFERED=1 会影响所有经 execute_train 提交的 Ray 作业，日志从批量缓冲改为逐行刷新，在高频打印场景下会带来轻微 IO 开销，一般可忽略。 - 影响：对用户（工程师）：训练、rollout 与转换作业的日志从“延迟一次性输出”变为实时可见，排障体验显著改善；对系统：所有 Ray 提交链路（train、p2p 权重传输、checkpoint 转换工具、eval 脚本）行为统一；对团队：这是 #1837 启动脚本加固系列的第一环，后续 PR #1896-#1911 的快照测试都建立在本次修正之上，错误拼写被测试永久拦截。 - 风险标记：全仓 75 文件机械替换，测试仅覆盖已跟踪的 Ray 启动器，无缓冲输出带来轻微 IO 开销

关联脉络

- PR #1896 Add a shell launch script test harness for future protection: 同一 #1837 跟踪系列的下一环，为 shell 启动脚本建立测试 harness，与本 PR 的启动器检查互为补充
- PR #1897 Fix various launch scripts errors about missing line concatenations or paths: 继续修复启动脚本错误（续行符与路径），与本 PR 处于同一批脚本加固工作流
- PR #1899 Snapshot the external commands of every shell launch script: 为全部 shell 启动脚本建立外部命令快照，覆盖的正是本 PR 批量修改的脚本集合
- PR #1901 Snapshot the commands and generated configs of every python launch script: 为 Python 启动脚本建立命令快照，验证 execute_train 生成命令的稳定性，与本 PR 修改的 command_utils 直接相关

- PR #1904 Move the shell exec helpers next to their only consumers: 后续重构 command_utils 与 exec_command 模块结构，与本 PR 核心文件相同