

PR #3289 完整报告

radixark/miles

Resolve file-run labels from the PR snapshot

合并时间: 2026-09-16 14:12

原文链接: <http://prhub.com.cn/radixark/miles/pull/3289>

执行摘要

- 一句话: 单文件 CI 改从 PR 快照解析标签白名单, 解除 /rerun-test 阻塞
- 推荐动作: 值得精读, 尤其是需要在 CI 中处理不可信输入的同学。重点看两处设计决策:
 - 一是 `_read_snapshot_labels()` 用 AST 字面量解析 + symlink 拒收 + 单次定义校验来“读快照而不 import”, 这是 fork PR 场景下读取配置的通用范式; 二是 `ci_register.py` 用默认参数注入白名单, 把行为变更收敛在 `file-run` 一条路径上, 最小化回归面。阅读时可一并追问三点: (a) 是否补 `tests/ci/test/test_file_run.py` 覆盖 symlink/ 重复定义 / 非字面量 dict 的错误分支; (b) `cwd` 契约是否改为显式传入 `labels_path` 更稳妥; (c) 是否需要在文档或注释里固化 `KNOWN_LABELS` 的写法约定。总体属于低风险、高收益的小修补, 可放心合并。

功能与动机

PR body 直接点明问题: "Single-file CI currently validates PR test registrations against the label registry on main. A label introduced by the PR therefore blocks every /rerun-test request during discovery, including requests for unrelated tests." 也就是 `collect_snapshot_tests()` 已经读 PR 快照的测试文件, 但 `RegistryVisitor` 内部写死模块级 `KNOWN_LABELS` (来自 main), 两者矛盾; 作者还引用了 #2849 的四个测试作为回归验证对象。放宽边界被明确限定为: "The resolver does not import PR modules; runner mappings and execution rules remain controlled by main. Unknown labels and malformed registries are still rejected."

实现拆解

1. 把标签白名单从模块常量改为可注入的实例状态 (`tests/ci/ci_register.py`):
`RegistryVisitor.__init__` 增加关键字参数 `known_labels=KNOWN_LABELS` 并保存为 `self.known_labels`; `_parse_call_args()` 中的 `label not in KNOWN_LABELS` 改为 `label not in self.known_labels`, 报错里的 `valid labels` 改用 `sorted(self.known_labels)`; `ut_parse_one_file()` 与 `collect_tests()` 逐层透传同名关键字参数。默认值保持 main 行为, 因此 fork CI image 的 `HANDLER.validate_source`、`run_suite` 等调用方零改动——这是本次只碰 3 个文件的前提。
2. 新增快照标签读取器 (`tests/ci/file_run.py:_read_snapshot_labels`): 以 `cwd` 下的 `tests/ci/labels.py` 为输入, 先用 `path.is_symlink()` 拒绝符号链接, 再 `ast.parse` 并在 `tree.body` 中匹配 `ast.Assign` 且 `target.id == "KNOWN_LABELS"` 的定义, 用 `len(definitions) != 1` 强制“只定义一次”, 最后 `ast.literal_eval` 取字面量并校验为 `str` ->

str dict; OSError / SyntaxError / ValueError / TypeError 统一包装为 FileRunError, 保证 main() 仍输出 ::error:: 并返回 1。新增的 import ast 属于 stdlib, 不违反该文件“只用 stdlib 与无依赖 registry 模块”的约束。

3. 在快照解析路径上接线 (tests/ci/file_run.py:collect_snapshot_tests) : 原 collect_tests(_discover_regular_ci_files(), sanity_check=True) 拆成两步, 改为 collect_tests(files, sanity_check=True, known_labels=_read_snapshot_labels()), 调用点在 os.chdir(root) 之后、finally: os.chdir(previous_directory) 之前, 确保读到的是快照而非 main 的注册表。
4. 执行计划仍由 main 决定: plan_file_run() 的 CUDA_SUITE_RUNS_ON、_IMAGE_TAG_PATTERN、disabled 校验与 container_image 拼接均未改动, 快照只贡献“标签名集合”这一项输入, 不参与 runner 选择与超时计算。
5. 测试配套 (tests/ci/test/test_fork_ci_image.py) : source fixture 在伪造 fork 快照里新增 tests/ci/labels.py, 首行是 raise RuntimeError('PR imports must not execute'), 第二行才是 KNOWN_LABELS: dict[str, str] = {'megatron': 'Megatron tests'}; 配合快照里的 tests/e2e/test_new.py (同样首行抛异常 + labels=['megatron'], hardware=['hopper']), test_validates_new_pr_registration_without_executing_source 同时证明“PR 新增标签可通过校验”和“整个解析过程没有 import PR 模块”。本次未新增 test_file_run.py 用例, _read_snapshot_labels 的 symlink、重复定义、非字面量 dict 等错误分支没有直接单测; PR body 给出的验证口径是 149 个既有 fork-CI/file-run/registry 测试 + pre-commit run --all-files 通过。

关键文件:

- tests/ci/file_run.py (模块 文件级重跑; 类别 infra; 类型 core-logic; 符号 _read_snapshot_labels, collect_snapshot_tests) : 新增 _read_snapshot_labels() 并在 collect_snapshot_tests() 中接线, 是本次修复的主体: 把 file-run 路径的标签白名单来源从 main 切换到 PR 快照, 同时用 AST 字面量解析守住“不 import 不可信代码”的底线。
- tests/ci/ci_register.py (模块 测试注册表; 类别 infra; 类型 core-logic; 符号 RegistryVisitor.init, RegistryVisitor._parse_call_args, ut_parse_one_file, collect_tests) : 注册表解析器的参数化改动: RegistryVisitor 与 ut_parse_one_file/collect_tests 新增 known_labels 关键字参数, 默认仍是 main 的常量, 从而让 file-run 能注入快照标签而不影响其他调用方。
- tests/ci/test/test_fork_ci_image.py (模块 镜像校验; 类别 test; 类型 test-coverage; 符号 source) : 测试配套: source fixture 在伪造 fork 快照中写入带 raise 语句的 tests/ci/labels.py, 使既有用例同时验证“新标签在快照内合法”和“解析过程不执行 PR 代码”。

关键符号: _read_snapshot_labels, collect_snapshot_tests, RegistryVisitor.init, RegistryVisitor._parse_call_args, ut_parse_one_file, collect_tests

关键源码片段

tests/ci/file_run.py

新增 _read_snapshot_labels() 并在 collect_snapshot_tests() 中接线, 是本次修复的主体: 把 file-run 路径的标签白名单来源从 main 切换到 PR 快照, 同时用 AST 字面量解析守住“不

import 不可信代码”的底线。

tests/ci/file_run.py: 单文件重跑解析器。该模块运行在未安装依赖的 hosted runner 上,
只能依赖 stdlib 与无依赖的 registry 模块; 入参是 fork 出来的不可信快照。

```
def _read_snapshot_labels() -> dict[str, str]:
    """从 PR 快照里的 tests/ci/labels.py 读取 KNOWN_LABELS, 只做数据解析, 不执行代码。

    这里刻意不用 import: 快照来自 fork PR, import 会把不可信仓库的模块级代码跑起来。
    """
    path = Path("tests/ci/labels.py") # 依赖调用方先 os.chdir 到快照根目录
    if path.is_symlink():
        # 拒收符号链接, 避免快照把注册表指向仓库之外的路径
        raise FileRunError(f"CI label registry must not be a symlink: {path}")
    try:
        tree = ast.parse(path.read_text(), filename=str(path))
        definitions = [
            node.value
            for node in tree.body
            if isinstance(node, ast.AnnAssign)
            and isinstance(node.target, ast.Name)
            and node.target.id == "KNOWN_LABELS"
        ]
        if len(definitions) != 1:
            # main 的写法是模块顶层恰好一次带注解的定义; 多写或漏写都视为注册表不可信
            raise FileRunError(f"{path} must define KNOWN_LABELS once as a literal dictionary")
        labels = ast.literal_eval(definitions[0]) # 只接受字面量, 杜绝表达式 / 函数调用求值
    # 任何读取或解析失败都收敛成 FileRunError, 由 main() 输出 ::error:: 并返回 1
    except (OSError, SyntaxError, ValueError, TypeError) as error:
        raise FileRunError(f"cannot read CI label registry {path}: {error}") from error
    if not isinstance(labels, dict) or not all(
        isinstance(key, str) and isinstance(value, str) for key, value in labels.items()
    ):
        raise FileRunError(f"{path}: KNOWN_LABELS must be a literal string-to-string dictionary")
    return labels
```

```
def collect_snapshot_tests(source_root: str | Path):
    """Parse a source snapshot as data, without importing its Python modules."""
    try:
        root = Path(source_root).resolve(strict=True)
    except OSError as error:
        raise FileRunError(f"cannot resolve CI source root {source_root}: {error}") from error
    if not root.is_dir():
        raise FileRunError(f"CI source root is not a directory: {root}")

    previous_directory = Path.cwd()
    try:
        os.chdir(root) # 切到快照根目录, _read_snapshot_labels 的相对路径才指向 PR 版本
```

```

files = _discover_regular_ci_files()
# 标签白名单来自 PR 快照；runner 映射与执行规则仍由 main 侧的 plan_file_run 掌控
return collect_tests(files, sanity_check=True, known_labels=_read_snapshot_labels())
finally:
    os.chdir(previous_directory)

```

tests/ci/ci_register.py

注册表解析器的参数化改动：RegistryVisitor 与 ut_parse_one_file/collect_tests 新增 known_labels 关键字参数，默认仍是 main 的常量，从而让 file-run 能注入快照标签而不影响其他调用方。

tests/ci/ci_register.py: 解析 register_cpu_ci / register_cuda_ci 等顶层调用并校验字段。

```

class RegistryVisitor(ast.NodeVisitor):
    # known_labels 默认取模块级 KNOWN_LABELS，也就是 main 分支上的标签注册表；
    # 只有 file-run 解析 PR 快照时才会显式覆盖为快照内的标签集合。
    def __init__(self, filename: str, *, known_labels=KNOWN_LABELS):
        self.filename = filename
        self.known_labels = known_labels
        self.registries: list[CIRegistry] = []

    def _parse_call_args(self, func_call: ast.Call, func_name: str) -> CIRegistry:
        # ... 位置参数、关键字参数、suite/est_time/nightly 等校验（本次未改动，省略）...

        labels = parsed.get("labels", [])
        backend = _REGISTER_BACKEND_MAP[func_name]
        if backend != HWBackend.CPU and not labels:
            raise ValueError(f"{self.filename}: labels in {func_name}() must contain at least one domain label")

        # 关键改动：白名单从“模块常量”变成“实例属性”，同一个解析器既能跑 main 的注册表，
        # 也能跑 PR 快照的注册表；未知标签依旧直接抛错，不会静默放过。
        unknown = [label for label in labels if label not in self.known_labels]
        if unknown:
            valid_list = ", ".join(sorted(self.known_labels))
            raise ValueError(
                f"{self.filename}: unknown labels {unknown} in {func_name}(); "
                f"valid labels: [{valid_list}]. "
                f"To add a new label: edit tests/ci/labels.py + create matching "
                f"`run-ci-<label>` in GitHub repo Settings -> Labels."
            )
        return CIRegistry(
            backend=backend,
            filename=self.filename,
            est_time=float(parsed["est_time"]),
            suite=parsed["suite"],
            labels=list(labels),
            hardware=list(parsed.get("hardware", [])),
            nightly=parsed.get("nightly", False),

```

```

        disabled=parsed.get("disabled", None),
        implicit=False,
    )

```

```

def ut_parse_one_file(filename: str, *, known_labels=KNOWN_LABELS) -> list[CIRRegistry]:
    with open(filename) as f:
        file_content = f.read()
    tree = ast.parse(file_content, filename=filename)
    # 关键字参数一路透传到访问器，默认值让既有调用方的行为完全不变
    visitor = RegistryVisitor(filename=filename, known_labels=known_labels)
    visitor.visit(tree)
    return visitor.registries

```

```

def collect_tests(files: list[str], sanity_check: bool = True, *, known_labels=KNOWN_LABELS) ->
list[CIRRegistry]:
    """解析每个 CI 文件；tags 白名单通过 known_labels 注入。"""
    ci_tests: list[CIRRegistry] = []
    for file in files:
        registries = ut_parse_one_file(file, known_labels=known_labels)
        # ... tests/fast/ 隐式 CPU 注册、sanity_check 等分支（本次未改动，省略）...
        ci_tests.extend(registries)
    return ci_tests

```

tests/ci/test/test_fork_ci_image.py

测试配套：source fixture 在伪造 fork 快照中写入带 raise 语句的 tests/ci/labels.py，使既有用例同时验证“新标签在快照内合法”和“解析过程不执行 PR 代码”。

tests/ci/test/test_fork_ci_image.py: source fixture 构造一份 fork 快照。

```

@pytest.fixture
def source(tmp_path, identity):
    root = tmp_path / "source"
    root.mkdir()

    def git(*args):
        return subprocess.check_output(["git", *args], cwd=root, text=True, stderr=subprocess.
DEVNULL).strip()

    git("init", "-q")
    git("config", "user.name", "Test")
    git("config", "user.email", "test@example.invalid")
    (root / "docker").mkdir()
    (root / "docker/Dockerfile").write_text("FROM scratch\n")
    (root / "docker/build.py").write_text("raise RuntimeError('untrusted driver must not execute')
\n")
    git("add", "docker")
    git("commit", "-qm", "base")

```

```

base = git("rev-parse", "HEAD")

(root / "docker/Dockerfile").write_text("FROM scratch\nLABEL pin=fixed\n")
# 本次新增：快照自带标签注册表，且首行就是会抛异常的语句；
# 只有走 AST 字面量解析才能通过，任何 import 都会先撞上 RuntimeError。
(root / "tests/ci").mkdir(parents=True)
(root / "tests/ci/labels.py").write_text(
    "raise RuntimeError('PR imports must not execute')\n"
    "KNOWN_LABELS: dict[str, str] = {'megatron': 'Megatron tests'}\n"
)
(root / "tests/e2e").mkdir(parents=True)
(root / "tests/e2e/test_new.py").write_text(
    "raise RuntimeError('PR imports must not execute')\n"
    "register_cuda_ci(est_time=1, suite='stage-b-2-gpu-h200', labels=['megatron'], hardware=
    ['hopper'])\n"
)
git("add", "docker/Dockerfile", "tests")
git("commit", "-qm", "fork")
head = git("rev-parse", "HEAD")
# 合成 merge commit 模拟 GitHub 的 refs/pull/N/merge，再把工作区切到该 commit
merge = git("commit-tree", "HEAD^{tree}", "-p", base, "-p", head, "-m", "merge")
git("checkout", "-q", "--detach", merge)
request, _, pr = copy.deepcopy(identity)
request.update(merge_sha=merge, inputs_hash=HANDLER.image_inputs.compute("HEAD",
root=root))
pr["head"]["sha"] = head
policy = resolve_workflow_inputs("pull_request", "", ["run-ci-image"])
return root, request, pr, policy

```

评论区精华

本 PR 没有任何人工 review 讨论：`comments_count` 与 `review_comments_count` 均为 0，唯一的 review 记录是 `claude[bot]` 的自动提示 "This repository is configured for manual code reviews. Comment `@claude review` for a one-time review..."，随后由作者 `yueming-yuan` 自行合并。因此可提炼的观点实际来自 PR 描述里的自我论证：①为什么要放宽——"A label introduced by the PR therefore blocks every /rerun-test request during discovery, including requests for unrelated tests."; ②放宽到什么程度——"The resolver does not import PR modules; runner mappings and execution rules remain controlled by main. Unknown labels and malformed registries are still rejected."; ③验证口径——149 个既有测试与 pre-commit 通过，#2849 的 4 个测试解析到预期 GPU suite 与 pr-2849 镜像，无需新增测试文件或重建镜像。结论：无争议点、无未决疑问，但这是一次有意的信任边界调整，缺少第二双眼睛复核。

- 快照标签解析的信任边界：读 PR 数据但不执行 PR 代码 (design)：设计被作者自行定稿并合入：标签名集合可以来自 PR 快照，但 runner 映射 (CUDA_SUITE_RUNS_ON)、镜像 tag 校验与执行计划仍完全由 main 决定，未知标签与畸形注册表继续报错。

风险与影响

- 风险：1) 信任边界被有意放宽：这是本次最实质的风险。file-run 路径上“标签合法性”的权威来源从 main 变为 PR 快照，`_read_snapshot_labels()` 会接受快照里任意自称 `KNOWN_LABELS` 的字面量集合；若快照把已有标签（如 `megatron`）重新定义或塞入 main 尚无的标签，校验都会放行。缓解在于 `plan_file_run()` 只使用 `suite / disabled / est_time`，runner 映射仍来自 main 的 `CUDA_SUITE_RUNS_ON`，因此不会串到别的 runner；但“未知标签必须报错”这一门禁在该路径上确实被弱化，需要在评审中显式接受。2) cwd 隐式契约：`_read_snapshot_labels()` 用相对路径 `Path("tests/ci/labels.py")`，正确性完全依赖 `collect_snapshot_tests()` 先 `os.chdir(root)`。任何后续重构（例如把读取挪到 `chdir` 之前、或让其他入口在仓库根调用它）都会静默读到 main 的注册表或抛 `OSError`→`FileRunError`，表现为 `/rerun-test` 莫名失败。当前没有断言或注释把这条不变量固化下来。3) 与 main 写法的强耦合：解析器要求 `KNOWN_LABELS` 是模块顶层恰好一次的 `AnnAssign` 且值为字面量 `dict`。若某个 PR 把 `tests/ci/labels.py` 改成普通赋值 `KNOWN_LABELS = {...}`、在函数里构造、或拆成多次赋值，`_read_snapshot_labels()` 会抛 `must define KNOWN_LABELS once as a literal dictionary`，该 PR 的 file-run 仍会失败——只是失败原因从“标签未知”换成了“注册表格式不符”。4) 错误分支缺少直测：`symlink` 拒收、`len(definitions) != 1`、非 `dict`、非 `str`→`str` 四类 `FileRunError` 分支没有对应单测（本次未改 `tests/ci/test/test_file_run.py`），回归保护依赖 `test_fork_ci_image.py` 的间接覆盖。5) 快照缺失文件：若 PR 快照里不存在 `tests/ci/labels.py`，`path.read_text()` 抛 `OSError` 被包装为 `FileRunError`，`/rerun-test` 直接报错退出（`fail-closed`，安全但不友好，错误信息会把路径带出来便于定位）。性能上无新增开销：每个 file-run 请求只多解析一个小文件。
- 影响：对贡献者：修复了“PR 一旦在 `tests/ci/labels.py` 里新增标签，该 PR 上所有 `/rerun-test` 请求（含无关测试）全部被阻塞”的死锁场景，#2849 中请求的 4 个测试现已可解析到预期 GPU suite 与 pr-2849 镜像。对 CI 系统：file-run 路径的标签白名单来源从 main 切换为 PR 快照，但 runner 选择、镜像 tag 校验、超时计算、执行规则仍由 main 侧决定，整体可控面没有被交给不可信代码；其余路径（fork CI image 校验、`run_suite` 收集）因默认参数保持 main 行为，无回归。对团队：`tests/ci/labels.py` 的结构（模块顶层单次 `AnnAssign` + 字面量 `dict`）由此变成跨 PR 的隐式契约，后续改动该文件时需注意；同时也提示类似“main 校验 PR 快照”的其他门禁点值得排查同类矛盾。
- 风险标记：信任边界有意放宽，cwd 隐式契约，与 `labels.py` 写法强耦合，错误分支缺少直测，无人工评审

关联脉络

- PR #2701 feat(ci): declare supported GPU generations on every CUDA test: 确立了 `tests/ci/ci_register.py` 上 `register_*_ci` 的 `labels/hardware` 校验契约（`KNOWN_LABELS`、`CUDA_STAGES`、`auto_arch` 等），本次正是在这套解析器上做参数化注入。
- PR #3234 fix(ci): prepare PR images only for selected GPU tests: 同样属于 `tests/ci` 的 PR 门禁 / 镜像准备策略调整，与本 PR 共同决定“PR 的快照如何映射到具体 GPU stage 与镜像”，是同一片 CI 策略面。

- PR #3184 ci: declare hardware on the two CUDA tests that landed after #2701: 说明注册表校验 (hardware/labels 声明) 是 CI 的高频破绽点; 本 PR 处理的是同类问题的另一面——标签白名单与 PR 快照不同步。
- PR #2876 ci(e2e): Harbor trials on real e2b sandboxes through the full rollout path (manual for now): 引入了注释命令 (/rerun-test 类) 与 tests/ci/labels.py 标签体系的文件级执行路径, 是本 PR 修复的 file-run 流程的直接上游。