

PR #3196 完整报告

radixark/miles

fix(ci): stop a CPU failure from skipping the GPU stages a nightly asked for

合并时间: 2026-09-11 03:14

原文链接: <http://prhub.com.cn/radixark/miles/pull/3196>

执行摘要

- 一句话: 修复 CI 门控: CPU 失败不再跳过 nightly 的全部 GPU stage
- 推荐动作: 值得精读, 但重点不是 22 行代码量, 而是两个可复用的工程经验: 一是 GitHub Actions 中「无 if 的 job 按整个依赖闭包判定 success()」这一反直觉语义导致的跨阶段级联跳过, 任何多阶段流水线都可能踩到; 二是把这类隐性约束固化成不变量测试的做法。建议进一步关注: 把断言从硬编码两个 job 扩展为对 GPU stage 依赖链的遍历式检查; 把 if:/always() 子串匹配升级为按行首正则或 YAML 解析, 消除注释误命中; 顺带排查 resolve-ci-policy、docker-build 等同类前置 job 是否都带显式 always() 保护。

功能与动机

PR body 明确指出问题: "Tonight's nightly (34493807332) ran no GPU test at all. Seven GPU stages were skipped, and not because of the policy" —— resolve-ci-policy 输出的 bypass_fastfail=true、skipped_stages=[] 完全正常。根因是 "resolve-ci-image carried no if, and a job without one is gated on the default success() across its whole dependency closure rather than on its direct needs", 而 stage-a-cpu 经 docker-build 进入了这个闭包。触发源是一次低级错误: tests/e2e/agent/ci/test_harbor_e2b_rollout.py 的 register_cuda_ci() 缺少 hardware= 参数, 使 collect_tests(sanity_check=True) 在建计划阶段就抛错; 由于 nightly 是唯一常规 GPU e2e 覆盖 ("PRs do not run GPU stages unless labelled"), 这次门控缺陷直接吃掉一整夜的 GPU 测试, 而 run 仍以普通部分失败上报, 掩盖了严重性。

实现拆解

1. 问题定位 (对照两次 nightly) : PR body 用 34368332492 (09-09, CPU 成功 → GPU 正常) 与 34493807332 (09-10, CPU 失败 → 全部 GPU skip) 做单变量对照, 证明差异仅来自 CPU 结果。关键机制是 GitHub Actions 的语义: 没有 if 的 job 不是只按 needs 判定, 而是按整个依赖闭包的 success() 判定; stage-a-cpu 经由 docker-build 传递进 resolve-ci-image 的闭包, docker-build 靠自身的 bypass_fastfail 逻辑走绿, resolve-ci-image 却把被放行的 fail-fast 又施加了回来。
2. 核心修复 (.github/workflows/pr-test.yml) : 给 resolve-ci-image 增加 if: always() && ! cancelled() && needs.docker-build.result == 'success', 与 docker-build 已有的写法保持同构。语义上做两件事: 一是把判定范围从「整个闭包」收敛为「docker-build 自身结果」, 二是保留原注释所强调的设计意图 (构建失败 / 被跳过仍要阻断 suite, 避免拿陈旧镜像

跑测试)。 `always()` 保证上游失败时该 job 仍会被调度以完成判定, `!cancelled()` 则在人工取消时不再启动。

3. 回归护栏 (tests/ci/test/test_stage_selection.py) : 新增模块级常量 `WORKFLOW` (指向 `github/workflows/pr-test.yml`) 与测试 `test_a_cpu_failure_cannot_skip_the_gpu_stages_a_nightly_asked_for`。测试直接读取工作流文本, 用 `re.findall(r"\n([a-z][a-z0-9-]*):\n(.*?)\n(=?\n[a-z][a-z0-9-]*:\n\n|Z)", workflow, re.S)` 把 2 空格缩进的 job 切成 `{job: block}`, 再遍历 (`"docker-build", "resolve-ci-image"`) 断言每块同时含 `if: & always()`; 删掉那一行修复即抛出 `AssertionError: resolve-ci-image inherits success() over the closure, defeating bypass_fastfail`。这是把「反直觉的 Actions 语义」写成硬约束, 防止后续新增 job 悄悄把闭包门控带回来。
4. 触发源的独立修复: 导致 CPU job 失败的是 `tests/e2e/agentic/test_harbor_e2b_rollout.py` 缺少 `hardware=` 参数 (#2701 将该参数变为必填), 已由 PR#3184 补齐; PR#3196 只处理门控语义, 不重复修测试声明。
5. 验证: `pytest tests/ci/test/` 结果为 `905 passed / 1 skipped / 0 failed`, `pre-commit run --all-files` 干净; 改动本身无法在 PR 内实跑 `nightly` 路径, 最终确认取决于下一次 `nightly` 的 GPU stage 是否照常执行。

关键文件：

- `.github/workflows/pr-test.yml` (模块 CI 工作流; 类别 infra; 类型 configuration; 符号 `resolve-ci-image`, `docker-build`) : 核心修复点: 为 `resolve-ci-image` 增加显式 `if`, 把门控从整个依赖闭包收敛到 `docker-build` 自身结果, 恢复 `nightly` 的 GPU stage 调度。
- `tests/ci/test/test_stage_selection.py` (模块 阶段选择; 类别 test; 类型 test-coverage; 符号 `test_a_cpu_failure_cannot_skip_the_gpu_stages_a_nightly_asked_for`, `WORKFLOW`) : 新增不变量测试: 直接断言工作流中 `docker-build` 与 `resolve-ci-image` 都带显式 `always()` 门控, 删掉修复即失败, 防止闭包门控被重新引入。

关键符号: test_a_cpu_failure_cannot_skip_the_gpu_stages_a_nightly_asked_for,
resolve-ci-image

关键源码片段

`.github/workflows/pr-test.yml`

核心修复点：为 resolve-ci-image 增加显式 if，把门控从整个依赖闭包收敛到 docker-build 自身结果，恢复 nightly 的 GPU stage 调度。

```
# pr-test.yml —— resolve-ci-image 的门控修复
resolve-ci-image:
```

```
# 只以这一次镜像构建的结果为准。注意：GitHub Actions 里没有 `if` 的 job 不是
# 按 `needs` 判定，而是按整个依赖闭包判定默认 success()，所以 stage-a-cpu 的
# 失败会经 docker-build 传导进来，连带跳过本 job 与全部 GPU stage —— 即使
# nightly 策略已设置 bypass_fastfail=true，其目的正是让 GPU suite 继续运行。
needs: [docker-build]

# always() && !cancelled(): 上游无关失败不再阻断，但显式取消时仍不启动；
# needs.docker-build.result == 'success': 镜像构建失败或被跳过时依旧阻断后续
# suite。避免拿陈旧镜像跑测试（保留原注释里「检测或构建失败必须停下、
```

```
# 而不是用陈旧镜像继续跑」的设计意图)。
if: always() && !cancelled() && needs.docker-build.result == 'success'
runs-on: ubuntu-latest
outputs:
  # 下游 GPU stage 从这里取镜像名，因此本 job 必须在 CPU 失败时仍能正常产出
  container_image: ${{ steps.resolve.outputs.container_image }}
```

tests/ci/test/test_stage_selection.py

新增不变量测试：直接断言 workflows 中 docker-build 与 resolve-ci-image 都带显式 always() 门控，删掉修复即失败，防止闭包门控被重新引入。

```
# 工作流文件路径从测试文件向上回溯三级定位到仓库根目录
WORKFLOW = Path(__file__).resolve().parents[3] / ".github/workflows/pr-test.yml"
```

```
def test_a_cpu_failure_cannot_skip_the_gpu_stages_a_nightly_asked_for():
```

```
    """bypass_fastfail 会被路径上任何继承默认 success() 的 job 击穿。
```

```
    没有 `if` 的 job 会按整个依赖闭包而非直接 needs 判定成功，因此在
    2026-09-10 的 nightly 里，一次 stage-a-cpu 失败跳过了 resolve-ci-image,
    并连带跳过全部七个 GPU stage —— 尽管 nightly 策略已设置
    bypass_fastfail=true，本意正是让这些 suite 继续运行。
    """
```

```
    workflow = WORKFLOW.read_text()
```

```
    # 用正则把 pr-test.yml 按 2 空格缩进的 job 名切成 {job: 块文本},
```

```
    # 不做 YAML 解析，因此 job 名必须只含小写字母、数字与连字符。
```

```
    blocks = dict(re.findall(r"\n ([a-z][a-z0-9-]*):\n(.*)?(?=\n [a-z][a-z0-9-]*:\n\Z)", workflow, re.
    S))
```

```
    # 这两个 job 是 GPU stage 的必经前置；任何一个退回默认 success(),
```

```
    # 都会让无关的 CPU 失败重新吃掉整晚的 GPU 覆盖。
```

```
    for job in ("docker-build", "resolve-ci-image"):
```

```
        block = blocks[job]
```

```
        assert "if:" in block, f"{job} inherits success() over the closure, defeating bypass_fastfail"
```

```
        assert "always()" in block, f"{job} must not be gated on an unrelated upstream failure"
```

评论区精华

本 PR 没有实质 review 讨论：review_comments_count = 0，claude[bot] 只留下手动触发 review 的模板评论 ("This repository is configured for manual code reviews. Comment @claude review for a one-time review")，guapisolo 给出 APPROVED 但 body 为空。因此设计权衡全部来自 PR body 与 commit message，最有价值的几点是：作者用单变量对照表把「门控闭包」这类隐性语义讲清楚；明确保留原设计意图而非简单删掉门控 ("That intent is right and is preserved; what it actually bought was 'any upstream failure stops the suites', including CPU tests that have nothing to do with the image")；以及把复现条件固化成不变量测试。没有第三方对该门控表达式做交叉验证，是这次审核的主要空缺。

- 为何保留门控而不是直接放开 GPU stage (design): 设计意图保留、作用域修正；! cancelled() 额外保证人工取消时不再启动该 job。

- 把 Actions 闭包门控语义固化为不变量测试 (testing): 以低成本文本断言替代 YAML 解析, 能防住本次回归, 但 job 名单硬编码、子串匹配可能被注释误命中。
- 审核缺少实质技术讨论 (other): 变更被合并, 但缺少第三方对 `always()/!cancelled()/result == 'success'` 组合的独立复核。

风险与影响

- 风险:
 1. 门控语义变更的边界: `always()` 会让 `resolve-ci-image` 在上游失败时依然被调度, 实际是否继续取决于 `needs.docker-build.result == 'success'`; 从 patch 上下文看该 job 的 `runs-on: ubuntu-latest` 与 `outputs.container_image` 只消费 `docker-build` 的结果, 未读取其它被 `skip` 的前置输出, 风险可控, 但若后续有人在其中引用 `stage-*-cpu` 的产物, `always()` 会掩盖空值问题。
 1. 测试断言偏脆弱 (`tests/ci/test/test_stage_selection.py`): `assert "if:" in block` 与 `assert "always()" in block` 都是子串匹配, 若 job 块内的注释或字符串里出现 `if:` 字面量, 即使真正缺 `if` 也可能误判通过; 正则 `\n ([a-z][a-z0-9-]*):\n` 还硬编码了「作业名只含小写字母 / 数字 / 连字符 + 恰好 2 空格缩进」, `pr-test.yml` 一旦调整缩进或引入非该命名规则的 job, `blocks[job]` 会 `KeyError` 形成假失败。
 2. 覆盖范围硬编码: 测试只遍历 (`"docker-build"`, `"resolve-ci-image"`) 两个名字, GPU stage 依赖链上新增的 `resolve-*` 前置 job 不会被自动纳入断言, PR body 中「下一个 job 无法悄悄复现闭包门控」的保证是局部的。
 3. 验证滞后: 修复正确性最终由下一次 `nightly` 证实, PR 阶段只有文本断言和单元测试, 属于「跑挂才知道」的隐性回归; 同时触发源 (缺 `hardware=`) 同类错误仍依赖 `register_cuda_ci()` 解析器在 `sanity_check=True` 时抛错兜底, 属被动防线。- 影响: 对用户与训练 / 推理主路径零影响, 全部改动位于 CI 基础设施与测试。对团队影响显著: `nightly` 是唯一常规 GPU e2e 覆盖, 修复前任何一次 CPU 侧失败都会静默吞掉一整夜 GPU 测试, 且 run 仍以普通部分失败上报, 容易被当成噪音忽略; 修复后 CPU 失败不再阻断 GPU suite, 构建类失败仍然 `fail-fast`, 门控行为与 `nightly` 策略的 `bypass_fastfail` 意图重新对齐。长期看, 新增的不变量测试把该语义缺口纳入 CI 自检, 降低同类事故复发概率, 代价是需要维护一个基于正则的工作流断言。- 风险标记: CI 门控语义变更, 工作流断言依赖文本正则, 断言覆盖硬编码两个 job, 缺少实质人工 review, 修复效果依赖下一次 `nightly` 验证

关联脉络

- PR #3184 ci: declare hardware on the two CUDA tests that landed after #2701: 正是本次事故的 CPU 失败触发源: 补齐 `tests/e2e/agent/test_harbor_e2b_rollout.py` 与 `tests/fast-gpu/test_true_on_policy_logprob_chunking.py` 缺失的 `hardware=` 参数, PR#3196 body 明确写 "Already fixed by #3184, twelve hours after the nightly"。
- PR #2701 feat(ci): declare supported GPU generations on every CUDA test: 将 `register_cuda_ci()` 的 `hardware=` 变为必填并引入注册解析器, 是本次 CPU 解析报错的直接前置; 其漏扫的文件直接导致了闭包门控被触发。

- PR #2876 ci(e2e): Harbor trials on real e2b sandboxes through the full rollout path (manual for now): 引入了缺少 hardware= 声明的 tests/e2e/agentc/test_harbor_e2b_rollout.py, 是本次级联跳过的起点文件。
- PR #3148 [CI] temporarily disable failing FT tests: 同属「CI 门控 / 阻塞」维护线: 都以临时或结构性手段解决 CI 红灯对后续阶段的连带影响, 可对比其应急处理与本 PR 的语义修复路径。
- PR #3152 ci(lark): explain each failed job on the CI status card: 同属 CI 失败可观测性脉络: 本次事故长期未被自动识别为「GPU 覆盖丢失」, 与 Lark 卡片失败分析能力互补。