

PR #3234 完整报告

radixark/miles

fix(ci): prepare PR images only for selected GPU tests

合并时间: 2026-09-15 06:04

原文链接: <http://prhub.com.cn/radixark/miles/pull/3234>

执行摘要

- 一句话: CPU-only PR 跳过 CUDA/ROCm 镜像准备
- 推荐动作: 值得精读。该 PR 展示了 CI 门控设计中「需求推导」与「复用现有 gate」的平衡, 尤其是 diff 缺失时保留 runnable 语义的修复, 以及提交历史中「先扩展输出、后回退为复用条件」的演进过程, 对理解 milos CI 策略的 stage 选择模型很有帮助。关注 tests/ci/ci_policy.py 的 _write_github_outputs 与 tests/ci/stage_selection.py 的 select_skipped_gpu_stages 两处核心逻辑即可。

功能与动机

PR body 明确指出这是 #2548 的 follow-up: Symptom 是 CPU-only PR 修改 docker/Dockerfile 时, 旧 CUDA workflow 在 CPU 成功后仍会进入 docker-build, ROCm 的 resolve-ci-image 也不检查选定 stage。Root Cause 有二: 一是 docker-build 和 ROCm resolve-ci-image 不参考最终 GPU stage 选择; 二是 select_skipped_gpu_stages 在 diff 不可用时绕过 runnable 选择。期望行为是 CPU-only 跳过两条镜像路径。

实现拆解

1. 新增 needs_cuda_image 输出推导: 在 tests/ci/ci_policy.py 的 _write_github_outputs 中, 用 bool(CUDA_STAGES.keys() - set(policy.skipped_stages)) 推导 CUDA 镜像需求, 从 tests/ci/hardware.py 引入 CUDA_STAGES。该输出仅在 pull_request 事件中依据最终 stage 选择有意义, schedule/workflow_dispatch/push 时 skipped_stages 为空, 所以恒为 true, 保持既有行为。
2. 修正 diff 缺失时的 stage 选择语义: tests/ci/stage_selection.py 的 select_skipped_gpu_stages 把 changed_files is None 的判断移到 _runnable_stages 计算之后, diff 不可用时返回 PR_GPU_STAGES - runnable (保留 runnable 选择), 而不是之前整体返回空元组 (等同于所有 GPU stage 都 runnable)。ci_policy.py 的 main 也移除了 else 分支, diff 缺失时仍执行 stage 选择。
3. CUDA workflow 门控: .github/workflows/pr-test.yml 的 resolve-ci-policy job 输出 needs_cuda_image, docker-build 的 if 条件新增 needs.resolve-ci-policy.outputs.needs_cuda_image == 'true'。CPU-only PR 时该值为 false, 构建链被跳过; resolve-ci-image 原有 needs: [docker-build] 和 docker-build.result == 'success' 依赖, 因此在 CUDA 构建被跳过时整个解析链也不会执行。

4. ROCm workflow 门控: `.github/workflows/pr-test-rocm.yml` 的 `resolve-ci-image` 新增 `needs: [resolve-ci-policy]` 和基于 `skipped_stages` 的 `if` 条件 (检查 `stage-c-4-gpu-mi350` 是否在 `skipped` 列表中), 复用现有 MI350 stage 选择条件, 不新增 `policy` 输出。
5. 测试与文档配套: `tests/ci/test/test_stage_selection.py` 新增 `test_policy_cli_image_demand_matches_selected_gpu_tests` 参数化矩阵 (覆盖 CPU-only/CUDA-only/ROCm-only/mixed 选择与 `valid/empty/malformed diff`)、`test_image_jobs_require_their_selected_gpu_stages` (断言 workflow 文本中的门控条件); `tests/ci/test/test_ci_policy.py` 的 `test_cli_appends_exact_github_outputs` 更新期望输出, 并新增 `test_non_pr_cli_preserves_image_preparation` 保证 `schedule/manual/release` 场景镜像准备不变; `docs/developer/ci/00-stage.md`、`02-docker-build.md`、`01-label.md` 同步更新镜像优先级与门控说明。

关键文件:

- `tests/ci/ci_policy.py` (模块 策略解析; 类别 `source`; 类型 `core-logic`; 符号 `_write_github_outputs, main, resolve_workflow_inputs`): 核心策略输出逻辑: 新增 `needs_cuda_image` 推导, 并在 `diff` 缺失时调整控制流以保留 `runnable` 选择, 是本次修复的策略侧核心。
- `tests/ci/stage_selection.py` (模块 阶段选择; 类别 `source`; 类型 `core-logic`; 符号 `select_skipped_gpu_stages`): 修复 `diff` 缺失时 `select_skipped_gpu_stages` 绕过 `runnable` 选择的问题, 是本次行为修正的关键控制流调整。
- `.github/workflows/pr-test.yml` (模块 流程编排; 类别 `infra`; 类型 `infrastructure`): CUDA 镜像构建链的门控改动: `docker-build` 依赖新增 `needs_cuda_image == 'true'` 条件, 是 CPU-only PR 跳过镜像构建的落地位置。

关键符号: `_write_github_outputs, select_skipped_gpu_stages, main`

关键源码片段

`tests/ci/ci_policy.py`

核心策略输出逻辑: 新增 `needs_cuda_image` 推导, 并在 `diff` 缺失时调整控制流以保留 `runnable` 选择, 是本次修复的策略侧核心。

```
def _write_github_outputs(policy: WorkflowPolicy, output_path: str) -> None:
    raw_labels = " ".join(policy.raw_labels)
    bypass_fastfail = str(policy.bypass_fastfail).lower()
    skipped_stages = json.dumps(policy.skipped_stages, separators=(",", ":"))
    # 镜像需求直接从最终 stage 选择推导: 只要还有任何 CUDA stage 未被跳过
    # (即该 stage 被选中且可运行), 就需要准备 CUDA 镜像。
    # CPU-only PR 的 skipped_stages 覆盖全部 CUDA_STAGES, 因此这里为 false。
    needs_cuda_image = str(bool(CUDA_STAGES.keys() - set(policy.skipped_stages))).lower()
    with open(output_path, "a", encoding="utf-8") as output:
        output.write(f"cadence={policy.cadence}\n")
        output.write(f"raw_labels={raw_labels}\n")
        output.write(f"bypass_fastfail={bypass_fastfail}\n")
        output.write(f"skipped_stages={skipped_stages}\n")
```

```

# 新增输出：下游 docker-build job 的 if 条件直接消费它
output.write(f"needs_cuda_image={needs_cuda_image}\n")
print(
    f"Resolved CI policy: cadence={policy.cadence} labels=[{raw_labels}] "
    f"bypass_fastfail={bypass_fastfail} skipped_stages={skipped_stages}"
)

```

tests/ci/stage_selection.py

修复 diff 缺失时 `select_skipped_gpu_stages` 绕过 runnable 选择的问题，是本次行为修正的关键控制流调整。

```

def select_skipped_gpu_stages(
    *,
    event_name: str,
    changed_files: tuple[ChangedFile, ...] | None,
    registrations: Iterable[CIRegistry],
    run_policy: RunPolicy,
    raw_labels: Iterable[str],
) -> tuple[str, ...]:
    """Return local GPU stage IDs that a workflow can safely skip."""
    # 非 PR 事件 (schedule / workflow_dispatch / push) 不做任何 stage 剪枝
    if event_name != "pull_request":
        return ()

    registrations = tuple(registrations)
    # 先计算 runnable：由 cadence / label / 硬件的可运行注册表决定
    runnable = _runnable_stages(registrations, run_policy)
    # diff 缺失（如 git diff 失败）时不再简单返回空集让所有 stage 视为可运行，
    # 而是只跳过「不可运行」的 stage，保留 label 显式选中的可运行 stage。
    # 这是对 #2548 的修正：diff 缺失不应让未请求的 GPU stage 变成 runnable。
    if changed_files is None:
        return tuple(sorted(PR_GPU_STAGES - runnable))
    affected = _affected_stages(changed_files, registrations, run_policy)
    affected.update(_explicit_scope_stages(registrations, run_policy, raw_labels, runnable))
    # 最终跳过集合 = 全部 PR GPU stage - (可运行 ∩ 受影响)
    return tuple(sorted(PR_GPU_STAGES - (runnable & affected)))

```

评论区精华

本 PR 没有人工 review 评论，只有 claude[bot] 的自动提示和 yueming-yuan 的批准。但提交历史显示了两轮重要设计权衡：第一版提交为 ROCm 也新增了镜像需求输出，随后（第二次提交）被回退，理由写在提交信息里：「ROCm only resolves an existing image and does not participate in the CUDA PR image build this change is meant to gate」。第三次提交最终选择让 ROCm 直接复用 MI350 的 `skipped_stages` 条件，而不是新增另一个 policy 输出——这是「复用现有 gate 而非扩展输出面」的典型决策，避免了 policy 输出膨胀。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在 CUDA docker-build 门控的回归：若 `needs_cuda_image` 推导与最终 stage 选择不一致（例如 `CUDA_STAGES` 与 `PR_GPU_STAGES` 集合未同步），可能出现 CUDA 测试被选择但镜像没有构建，导致 GPU 测试卡在镜像等待。另一个风险是 diff 缺失路径的语义变化：现在 diff 不可用时 `skipped_stages` 不再是空集，而由 `_runnable_stages` 决定，若某 PR 的 label 选择包含大量 CUDA stage 且 diff 缺失，镜像准备仍会执行，但没有功能退化。测试覆盖方面，`test_image_jobs_require_their_selected_gpu_stages` 用字符串匹配断言 workflow 内容，workflow 结构调整（如 job 重命名）会导致测试脆弱但不会漏报。
- 影响：影响面是仓库的 PR CI 流程：所有 PR 触发时，`resolve-ci-policy` 额外输出 `needs_cuda_image`；CPU-only PR（无 GPU label、无 GPU 测试文件变更）将不再触发 CUDA 镜像构建和 ROCm 镜像解析，可节省 docker build 的 runner 时间和镜像存储。对开发者的可见影响是修改 docker/Dockerfile 的 CPU-only PR 不再获得镜像构建验证，需要主动加 `run-ci-image` 等 label 才触发，这一点已在 PR body 和文档中明确。对系统层面，减少了 CI 资源的无效消耗，属于正向优化。
- 风险标记：CI 门控变更，依赖集合一致性，diff 缺失路径语义变化

关联脉络

- PR #2548 feat(ci): declare supported GPU generations on every CUDA test: PR body 明确本 PR 是 #2548 的 follow-up，修复其引入的 CPU-only PR 仍进入镜像准备的问题。
- PR #3196 fix(ci): stop a CPU failure from skipping the GPU stages a nightly asked for: 同为 CI 门控 bugfix，涉及 GPU stage 选择与 gate 的语义修正，属于同一 CI 策略演进线。
- PR #2701 feat(ci): declare supported GPU generations on every CUDA test: 为本 PR 使用的 `CUDA_STAGES` 注册体系奠定了基础，且是 #2548 的正式编号。