PR #2221 完整报告

radixark/miles

fix(ci): split CPU and GPU reusable workflows

合并时间：2026-08-07 04:00

原文链接：http://prhub.com.cn/radixark/miles/pull/2221

执行摘要

- 一句话：拆分 CPU/GPU CI 可复用工作流，消除冗余 skipped job
- 推荐动作：值得快速浏览。本 PR 展示了 GitHub Actions 可复用工作流拆分的一个干净范式：用“一个工作流只声明本硬件一个 job”替代“单工作流 + job 级 if”，从根上消除展开语义导致的冗余 skipped job；同时用结构锁测试固定 job id、路由计数与 cpu_runner 残留，防止回归。关注 _run-cpu-ci.yml 与 _run-ci.yml 的步骤对等性，以及 test_run_suite.py 中基于正则提取 job id 的断言方式——这是用测试守护 YAML 结构的好例子。

功能与动机

PR body 描述症状与根因：每个硬件 stage 都把另一硬件的 job 暴露为 skipped；根本原因是 _run-ci.yml 同时声明 run 与 run-cpu，而 jobs..if 在两个可复用 job 展开后才求值。修复还保留了另一层动机：CPU 快速测试继续跑在 GitHub-hosted ubuntu-latest 上，不占用 GPU fleet runner 槽位；同时 job 列表变干净，方便定位真正执行的任务。commit message 也明确说明要保留 runner、command、secret、gating 与 check-name 契约。

实现拆解

变更入口是 pr-test.yml 的 stage 路由，核心是工作流文件拆分，并配套结构锁测试与文档更新。

1. 新建 .github/workflows/_run-cpu-ci.yml：把原 _run-ci.yml 中的 run-cpu job 整体迁出，job id 保持 run-cpu，runner 仍是 ubuntu-latest。文件开头注释记录了拆分动机：GitHub 会先展开可复用工作流内的所有 job 再评估 if，同文件双 job 必然产生跨硬件 skipped 兄弟。迁移内容包含 resolve-refs 依赖 ref 解析（支持 PR body 的 ci-megatron-pr / ci-sglang-pr 指令与 refs/pull/#N/head 转换）、sglang 与 Megatron-LM checkout、PYTHONPATH 配置、requirements-ci-cpu.txt 裸 pip 安装、verify_source_resolution.py 校验、--list-only 计划打印与 HF_TOKEN 注入。workflow_call 输入只保留 execute_command。
2. 精简 .github/workflows/_run-ci.yml：删除 cpu_runner input、run job 的 if 条件以及整个 run-cpu job，runs_on 描述同步去掉 cpu_runner: false 字样。文件现在只承担 GPU 路径，run job id 不变，既有 gate 与 check-name 契约不受影响。
3. 调整 pr-test.yml 路由：stage-a-cpu 与 stage-b-cpu 的 uses 改为 ./.github/workflows/_run-cpu-ci.yml 并删除 cpu_runner: true 传参；GPU 五个 stage 仍

指向 _run-ci.yml 并继续传递 runs_on 与 container_image。CPU 与 GPU 的 gating（needs: [resolve-ci-policy, resolve-ci-image]、CPU 失败短路 GPU）原样保留。

4. 测试配套：tests/ci/test/test_run_suite.py 新增 _reusable_workflow 辅助与 test_cpu_and_gpu_stages_use_dedicated_reusable_workflows。断言 pr-test.yml 中 _run-cpu-ci.yml 恰好出现 2 次、_run-ci.yml 恰好出现 5 次；再用正则提取两个工作流 jobs 下的 job id，断言 GPU 工作流只有 run、CPU 工作流只有 run-cpu，并且 cpu_runner 开关彻底消失。这把 job 结构契约锁死，防止 stage 增删或开关回退导致问题复发。
5. 文档配套：docs/ci/00-stage.md 更新 Runner selection 与 Launch 段落，说明 CPU stage 调用 _run-cpu-ci.yml、CUDA stage 调用 _run-ci.yml，每个工作流只声明本硬件 job，GitHub 不再为另一硬件追加 skipped 兄弟。验证方面，tests/ci/test/test_run_suite.py 本地 70 个测试通过，Ruff、check-yaml、git diff --check 与 commit hooks 均通过。

关键文件：

- .github/workflows/_run-cpu-ci.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：新文件承载全部 CPU 测试步骤，是本 PR 拆分方案的核心载体；从 _run-ci.yml 迁移 run-cpu job，保证 CPU stage 不再展开 GPU 兄弟 job。
- .github/workflows/_run-ci.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 run）：删除 cpu_runner 开关与 run-cpu job，使文件只留下 GPU 路径，是拆分方案的另一半。
- tests/ci/test/test_run_suite.py（模块 CI 测试；类别 test；类型 test-coverage；符号 _reusable_workflow, test_cpu_and_gpu_stages_use_dedicated_reusable_workflows）：新增锁测试，断言 pr-test.yml 的路由数量与每个可复用工作流的 job 结构，防止回归。
- .github/workflows/pr-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure；符号 stage-a-cpu, stage-b-cpu）：入口路由变更：两个 CPU stage 改调 _run-cpu-ci.yml，移除 cpu_runner 传参；决定整个拆分是否生效。
- docs/ci/00-stage.md（模块 CI 文档；类别 docs；类型 documentation）：同步 CI 文档，说明 Runner 选择与 Launch 语义变化，避免文档与实现脱节。

关键符号：_reusable_workflow, test_cpu_and_gpu_stages_use_dedicated_reusable_workflows

关键源码片段

tests/ci/test/test_run_suite.py

新增锁测试，断言 pr-test.yml 的路由数量与每个可复用工作流的 job 结构，防止回归。

```
@staticmethod
def _reusable_workflow(name: str) -> str:
    # 读取 .github/workflows 下的可复用工作流，供结构锁测试使用
    return (
        Path(__file__).resolve().parents[3] / ".github" / "workflows" / name
    ).read_text()
```

```
def test_cpu_and_gpu_stages_use_dedicated_reusable_workflows(self):
    # 锁住 pr-test.yml 的路由：CPU 两个 stage 走 _run-cpu-ci.yml，GPU 五个 stage 走 _run-ci.
    yml
    workflow = self._workflow()
    assert workflow.count("uses: ./.github/workflows/_run-cpu-ci.yml") == 2
    assert workflow.count("uses: ./.github/workflows/_run-ci.yml") == 5
    assert "cpu_runner" not in workflow

    # 每个可复用工作流只能声明本硬件对应的一个 job，
    # 否则 GitHub 展开后仍会给另一硬件 stage 追加 skipped 兄弟 job
    gpu_workflow = self._reusable_workflow("_run-ci.yml")
    cpu_workflow = self._reusable_workflow("_run-cpu-ci.yml")
    job_id_pattern = r"^  ([A-Za-z_][A-Za-z0-9_-]*):$"
    gpu_jobs = re.findall(job_id_pattern, gpu_workflow.split("\njobs:\n", 1)[1], re.MULTILINE)
    cpu_jobs = re.findall(job_id_pattern, cpu_workflow.split("\njobs:\n", 1)[1], re.MULTILINE)
    assert gpu_jobs == ["run"]
    assert cpu_jobs == ["run-cpu"]
    # 迁移后应彻底移除 cpu_runner 开关，避免新旧双路径并存
    assert "cpu_runner" not in gpu_workflow
    assert "cpu_runner" not in cpu_workflow
```

评论区精华

本 PR 没有任何 review 评论，yushengsu-thu 以空 body 直接 APPROVED，因此没有产生争议线程。有效讨论信息浓缩在 commit message 中：GitHub expands every job in a reusable workflow before evaluating job-level conditions，即根因是可复用工作流的展开语义早于 job 级 if 求值。评审通过即认可该诊断与“一硬件一工作流”的拆分方向。测试承担了行为一致性论证：新锁测试从 YAML 文本层面断言路由计数与 job 单一性。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险来自双工作流复制带来的行为漂移。resolve-refs 依赖解析、依赖安装、verify_source_resolution.py 与 HF_TOKEN 注入等步骤现在同时存在于 _run-cpu-ci.yml 与 _run-ci.yml，后续若只修改一处，CPU 与 GPU 测试环境会逐渐不一致，而锁测试只检查 job id 与 uses 计数，不校验步骤内容，步骤级漂移无法被 CI 拦截。其次，路由计数被锁测试固定为 2 次 CPU、5 次 GPU，未来增减 stage 必须同步更新断言，否则 PR 会被测试拦住（这也是设计意图）。风险仅限 CI 工作流，不影响训练、rollout、模型等运行时路径；若 CPU job 迁移时遗漏某个步骤，通常表现为 CPU 测试环境显式报错，而不是静默失败。
- 影响：对 PR 提交者：GitHub 展开的 job 列表不再混入另一硬件的 skipped 兄弟 job，可读性和可排查性明显提升。对 CI 维护者：引入第二个专用可复用工作流，公共逻辑修改时需要同步两处。对算力：CPU 快速测试仍运行在 GitHub-hosted ubuntu-latest，不占用 GPU fleet runner 槽位；CPU 失败短路 GPU 的 gating 语义完全保留。对仓库产品代码：零影响。

- 风险标记：双工作流步骤复制，存在漂移风险，锁测试仅覆盖结构不校验步骤内容，stage 数量变化需同步更新锁测试，CPU/GPU 工作流行为一致性依赖手工维护

关联脉络

- PR #2231 fix(ci): cancel PR tests after closure: 同样修改 pr-test.yml 与 tests/ci/test/test_run_suite.py，同属 PR Test 工作流护栏体系。
- PR #2205 ci: authenticate CPU Hugging Face downloads: 在 _run-ci.yml 给 CPU 测试注入 HF_TOKEN，本 PR 将 CPU job 迁至 _run-cpu-ci.yml 时同步保留该 secrets 配置，两条变更在同一 CPU 测试路径上衔接。
- PR #2217 test(ci): right-size session model GPU coverage: 同一时段对 PR Test GPU stage 的规格调整，与本 PR 共同塑造 stage 的 runner 分配策略。