# PR #2674 完整报告

radixark/miles

[AMD] Let the ROCm suite honour the ci-sglang-pr and ci-megatron-pr directives

合并时间：2026-08-21 15:13

原文链接：http://prhub.com.cn/radixark/miles/pull/2674

## 执行摘要

- 一句话：ROCm CI 支持按 PR 指令覆盖镜像内置依赖
- 推荐动作：值得精读，重点是两个 workflow 文件的设计：用独立 resolver 作业做门控与布尔透传；覆盖策略“只替换命名的依赖，未命名保持镜像基线”避免意外漂移；Megatron patch 三状态探测与 fail-closed 保护。若后续要扩展 PR 指令体系，可参考此 PR 的 seam 测试与验证方法（scratch 提交 + 空提交重触发），以及 PR body 指令跨工作流消费的副作用教训。

## 功能与动机

PR body 明确说明问题："The ROCm suite parsed ci-sglang-pr: and ci-megatron-pr: directives but always kept dependency installation disabled, so a requested ref never reached an MI350 job." 也就是说解析逻辑早已存在，但门控开关从未被调用方打开，导致 AMD/ROCm CI 无法按需验证新的 SGLang 或 Megatron-LM 版本，只能依赖镜像内置版本。

## 实现拆解

实现按以下步骤拆解：

1. 入口门控：在 .github/workflows/pr-test-rocm.yml 新增 resolve-ci-deps 作业，读取 github.event.pull_request.body 与 workflow_dispatch 输入（ci_megatron_pr / ci_sglang_pr），用 grep -qP '^ci-(megatron|sglang)-pr:\s+\S+' 判断是否点名依赖 ref，并输出 skip_dependency_install 布尔标志；stage-c-4-gpu-mi350 的 needs 增加该作业，在 if 条件中校验其 result == 'success'，同时把输出作为参数转发给可复用工作流。
2. 条件化安装：在 .github/workflows/_run-ci-rocm.yml 删除原来“未命名依赖回退到移动分支（miles-main / sglang-miles）”的逻辑，改为对 SGLANG_PR 与 MEGATRON_PR 各自的 if [ -n ... ] 独立安装；这样只替换 PR 实际命名的依赖，未命名的一方保留 rocm/sgl-dev 镜像内置版本，避免单条指令意外漂移另一份依赖。
3. Megatron patch 重放：Megatron ref checkout 后，对 $GITHUB_WORKSPACE/docker/amd_patch/latest/megatron.patch 依次执行 git apply --check、git apply --reverse --check 与失败分支：可应用则应用、已包含则跳过、否则输出 ::error 并 exit 1；patch 必须从 checkout 的 Miles workspace 读取，因为镜像在运行时已删除 /tmp/amd_patch。

4. 测试与文档配套：tests/ci/test/test_run_suite.py 中 test_stage_consumes_policy_and_preserves_manual_full_scope 增加对 resolve-ci-deps 依赖链与 skip_dependency_install 转发的断言，新增 test_megatron_override_preserves_rocm_patch 验证 Bash 片段中 checkout → patch check → apply → reverse-check → error → fail → install 的先后顺序；docs/ci/00-stage.md 同步更新 ROCm 依赖边界说明（命名单侧覆盖，未命名保持镜像内置）。
5. 验证方式：作者先静态核对 main 上 skip_dependency_install 默认值与门控位置，再通过两次真实 MI350 运行（一次 workflow_dispatch 输入、一次 PR body 指令配合临时 scratch 提交强制启用 stage）证明 flag 到达 install 步骤；scratch 提交随后回滚，test_run_suite.py 全套 87 个测试通过。

关键文件：

- .github/workflows/pr-test-rocm.yml（模块 CI 编排；类别 infra；类型 infrastructure；符号 resolve-ci-deps job, skip_dependency_install output, ci_megatron_pr input, ci_sglang_pr input）：新增 resolve-ci-deps 门控作业，从 PR body 或 dispatch 输入识别依赖指令，输出 skip_dependency_install 并接入 MI350 stage 的依赖链与参数转发，是整个功能生效的入口。
- .github/workflows/_run-ci-rocm.yml（模块 CI 编排；类别 infra；类型 infrastructure；符号 SGLANG conditional install, Megatron patch three-state probe, skip_dependency_install input）：实现核心覆盖逻辑：只有命名的依赖被 fetch/install，未命名保持镜像基线；并为 Megatron 覆盖重放 ROCm 兼容 patch。
- tests/ci/test/test_run_suite.py（模块 运行套件；类别 test；类型 test-coverage；符号 test_megatron_override_preserves_rocm_patch, test_stage_consumes_policy_and_preserves_manual_full_scope）：守护 workflow seam 契约：更新既有断言以覆盖 resolve-ci-deps 依赖链与 skip_dependency_install 转发，并新增 test_megatron_override_preserves_rocm_patch 验证 Bash 覆盖脚本的关键顺序。
- docs/ci/00-stage.md（模块 CI 文档；类别 docs；类型 documentation）：同步更新依赖边界说明，明确 ROCm 阶段默认使用镜像内置版本、仅在命名单侧时覆盖，帮助维护者理解新行为。

关键符号：resolve-ci-deps (workflow job), test_megatron_override_preserves_rocm_patch, test_stage_consumes_policy_and_preserves_manual_full_scope, stage-c-4-gpu-mi350 (workflow job)

## 关键源码片段

### .github/workflows/pr-test-rocm.yml

新增 resolve-ci-deps 门控作业，从 PR body 或 dispatch 输入识别依赖指令，输出 skip_dependency_install 并接入 MI350 stage 的依赖链与参数转发，是整个功能生效的入口。

```
  # resolve-ci-deps 是 ROCm 套件的入口门控：判定 PR body 或 dispatch 输入
  # 是否点名依赖 ref，再把 skip_dependency_install 布尔结果输出给 MI350 stage。
```

```
resolve-ci-deps:
  runs-on: ubuntu-latest
  outputs:
    skip_dependency_install: ${{ steps.resolve.outputs.skip_dependency_install }}
  steps:
    - name: Resolve whether the image's baked dependencies are overridden
      id: resolve
      shell: bash
      env:
        PR_BODY: ${{ github.event.pull_request.body || '' }}
        INPUT_MEGATRON_PR: ${{ github.event.inputs.ci_megatron_pr || '' }}
        INPUT_SGLANG_PR: ${{ github.event.inputs.ci_sglang_pr || '' }}
      run: |
        NAMED_BY=""
        if [ -n "$INPUT_MEGATRON_PR" ] || [ -n "$INPUT_SGLANG_PR" ]; then
          NAMED_BY="the dispatch inputs"
        elif [ -n "$PR_BODY" ] && echo "$PR_BODY" | grep -qP '^ci-(megatron|sglang)-pr:\s+\S+';
        then
          NAMED_BY="the pull request body"
        fi
        # 未点名任何依赖时保持镜像内置版本，跳过安装步骤。
        if [ -n "$NAMED_BY" ]; then
          echo "skip_dependency_install=false" >> "$GITHUB_OUTPUT"
        else
          echo "skip_dependency_install=true" >> "$GITHUB_OUTPUT"
        fi
```

## .github/workflows/_run-ci-rocm.yml

实现核心覆盖逻辑：只有命名的依赖被 fetch/install，未命名保持镜像基线；并为 Megatron 覆盖重放 ROCm 兼容 patch。

```
# 只有命名的依赖才会被替换；未命名的一方保持 rocm/sgl-dev 镜像内置版本。
if [ -n "$SGLANG_PR" ]; then
  cd /sgl-workspace/sglang && git reset --hard HEAD && git clean -fd \
    && git fetch origin "$SGLANG_PR" && git checkout -f FETCH_HEAD \
    && git log --oneline -1 \
    && pip install -e python --no-deps --break-system-packages
fi
if [ -n "$MEGATRON_PR" ]; then
  cd /root/Megatron-LM
  git reset --hard HEAD
  git clean -fd
  git fetch origin "$MEGATRON_PR"
  git checkout -f FETCH_HEAD
  git log --oneline -1
  # ROCm 兼容 patch 三状态探测：能应用则应用，已包含则跳过，否则 fail-closed。
  # patch 必须从 checkout 的 Miles workspace 读取，镜像运行时已删除 /tmp/amd_patch。
  if git apply --check "$GITHUB_WORKSPACE/docker/amd_patch/latest/megatron.patch"; then
    git apply "$GITHUB_WORKSPACE/docker/amd_patch/latest/megatron.patch"
```

```
  elif git apply --reverse --check "$GITHUB_WORKSPACE/docker/amd_patch/latest/megatron.
  patch"; then
    echo "Megatron ROCm patch already present"
  else
    echo "::error::Selected Megatron ref is incompatible with the ROCm patch"
    exit 1
  fi
  pip install -e . --no-deps --break-system-packages
fi
cd "$GITHUB_WORKSPACE" && pip install -e . --no-deps --break-system-packages
```

## 评论区精华

核心讨论集中在三处：

- fzyzcjy 通过两轮实机验证说明“gate 是真实的”：第一轮用 workflow_dispatch 输入路径，第二轮在 PR 描述中写 ci-sglang-pr: c447264e... 并借助临时 scratch 提交让 MI350 阶段真正执行，验证完成后全部清理回滚。
- guapisolo 反馈在 PR 描述中添加 ci-sglang-pr: #1234 会导致 CPU CI 失败，随后从 PR 描述中移除该行，并发布空提交（04c1c550）以触发新的 pull_request 事件，让 CI 在修正后的 body 上重跑。
- guapisolo 相继合入两个关键修正：一是覆盖 Megatron ref 后重放 ROCm 兼容 patch 的三状态探测（applied / already-present / incompatible fail-closed）；二是把 patch 来源从 /tmp/amd_patch 改为 checkout 的 Miles workspace，因为镜像运行时会删除 /tmp/amd_patch。
- PR body 中 ci-sglang-pr: #1234 导致 CPU CI 失败 (correctness): PR body 指令会被多个工作流消费，验证用的指令行会干扰 CPU 套件；最终通过从描述移除解决，不涉及代码修改。
- 两轮实机验证证明 flag 到达 install step (testing): 确认 skip_dependency_install 门控真实生效、PR body 指令路径可用，验证过程全部清理。
- Megatron 覆盖时保留 ROCm 兼容 patch (design): patch 从 $GITHUB_WORKSPACE/docker/amd_patch/latest/megatron.patch 读取，保留 apply、already-present、incompatible 三种行为。

## 风险与影响

- 风险：风险点如下：
  - 指令解析仅匹配 ^ci-(megatron|sglang)-pr:\s+\S+ 行首格式，PR body 中列表项或缩进写法则不会被识别，可能造成“写了指令但不生效”的误判。
  - 依赖 ref 只能从镜像自身 checkout 的 origin 拉取，只存在于其他 remote 的 commit 或 branch 无法 resolve，workflow 注释已明确该边界。
  - Megatron 覆盖对不兼容 ref 采取 fail-closed（exit 1），虽然保护了 ROCm 兼容性，但会阻塞想测试任意分支的开发者，需要维护者知晓这一预期行为。

- PR body 指令会被多个工作流消费（CPU CI 已观察到失败案例），在 PR 描述中临时加指令验证时容易误伤其他套件。
- 改动集中在 CI workflow 与测试文档，不触碰训练与推理运行时代码，生产回归风险低。
- 影响：对 AMD/ROCm 开发者：可在 PR 中直接验证新 SGLang 或 Megatron ref，不再需要等待镜像重建；未点名任何依赖时行为与之前完全一致（skip_dependency_install=true）。对 CI 系统：pr-test-rocm.yml 多了一个 resolver 作业与输入转发链路，_run-ci-rocm.yml 的安装步骤更复杂，后续维护依赖 seam 测试守护。对团队工作流：文档明确了“镜像即基线，覆盖仅限命名侧”的运维模型，降低对 ROCm 镜像构建节奏的依赖。影响面仅限 .github/workflows、tests/ci 与 docs/ci，无用户可见功能变化。
- 风险标记：PR body 指令跨工作流误伤，Megatron patch 冲突 fail-closed，指令匹配仅限行首，依赖 ref 仅限镜像 origin

# 关联脉络

- PR #2671 [AMD] Enable four CI tests on ROCm: 同为 AMD/ROCm CI 扩展方向，本 PR 进一步让 ROCm 套件支持按需覆盖依赖 ref。
- PR #2670 fix(docker): update torch_memory_saver for CUDA VMM granularity: 同处 ROCm 镜像依赖管理上下文，本 PR 则提供运行期按需覆盖依赖的通道。
- PR #2660 [AMD] Point the ROCm image at the v0.5.16 wheels release with TE 2.17.0: ROCm 镜像依赖与构建策略演进，与本 PR 的依赖覆盖能力互补。
- PR #2600 fix(docker): pin cutlass-dsl 4.6.2 and flashinfer 0.6.15.post1 over the sglang base: 镜像内置依赖管理（pin 与覆盖）的同一技术脉络。
- PR #2496 feat(ci): add authorized comment-to-label gateway: PR 指令 / 标签体系的一部分，ci-sglang-pr: / ci-megatron-pr: 指令同属 PR 指令生态。