PR #23492 完整报告

sgl-project/sglang

[CI] /rerun-stage: auto-include wheel build when PR modifies sgl-kernel/

合并时间：2026-04-23 02:28

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23492

执行摘要

- 一句话：修复 /rerun-stage 命令在修改 sgl-kernel 的 PR 上错误使用 PyPI wheel 的问题，自动包含内核构建。
- 推荐动作：建议 CI 维护者和经常使用 /rerun-stage 命令的工程师仔细阅读此 PR，以理解内核构建的自动包含机制。关注 scripts/ci/utils/slash_command_handler.py 中的变更逻辑和 .github/workflows/pr-test.yml 中的条件调整，这些设计决策确保了向后兼容性和安全性，对于类似 CI 流程改进有参考价值。

功能与动机

在调试 PR #21247（torch 2.11 升级）时发现，/rerun-stage stage-c-test-deepep-4-gpu-h100 在修改 sgl-kernel/ 的 PR 上会静默使用 PyPI sgl-kernel wheel 而不是 PR 的更改，因为 sgl-kernel-build-wheels 在 target_stage 模式下无条件跳过。这导致测试没有正确执行 PR 分支的变更，掩盖了潜在问题。

实现拆解

1. 命令处理器增强：在 scripts/ci/utils/slash_command_handler.py 的 handle_rerun_stage 函数中，调用 has_sgl_kernel_changes(pr) 助手检测 PR 是否修改 sgl-kernel/。如果检测到变更且目标阶段是 NVIDIA（非 AMD），则设置 include_wheel_build: "true" 到工作流输入，并确保 pr_head_sha 被传递以支持后续检测。
2. 工作流配置调整：在 .github/workflows/pr-test.yml 中，添加新的 workflow_dispatch 输入 include_wheel_build（布尔值，默认 false）。调整 sgl_kernel 输出逻辑，使其在 target_stage 模式下且 include_wheel_build 为 true 时保留真实值，允许 sgl-kernel-build-wheels 作业运行。同时更新 validate-target-stage 步骤，仅在 include_wheel_build 未设置时失败，提供安全网。
3. 构建作业条件优化：修改 sgl-kernel-build-wheels 和 sgl-kernel-build-wheels-arm 作业的 if: 条件，允许在 target_stage 模式下且 include_wheel_build 为 true 时运行，使用 always() 和显式结果检查以避免阻塞。
4. 目标阶段依赖更新：在 stage-c-test-* 作业中添加 sgl-kernel-build-wheels 到 needs 列表，确保目标阶段等待新构建的 wheel 下载，尽管在非 target_stage 模式下已有传递依赖。
5. 测试与验证：PR body 中提供了详细的测试计划，包括单元测试和工作流场景验证，确保变更正确且无回归。

关键文件：

- scripts/ci/utils/slash_command_handler.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 handle_rerun_stage）：处理 /rerun-stage 命令的入口点，负责自动检测内核变更并设置构建标志，是 CI 流程的核心逻辑变更。
- .github/workflows/pr-test.yml（模块 工作流配置；类别 infra；类型 infrastructure）：定义 CI 工作流的主要配置文件，添加新输入并调整逻辑以支持内核构建在 target_stage 模式下运行，影响整个测试流程。

关键符号：handle_rerun_stage

关键源码片段

scripts/ci/utils/slash_command_handler.py

处理 /rerun-stage 命令的入口点，负责自动检测内核变更并设置构建标志，是 CI 流程的核心逻辑变更。

```
def handle_rerun_stage(pr, stage_name, is_amd_stage=False):
    # 检查 PR 是否修改了 sgl-kernel/ 目录
    kernel_changes = has_sgl_kernel_changes(pr)
    if kernel_changes:
        print(
            "PR modifies sgl-kernel/ - setting include_wheel_build=true so the "
            "target stage gets the freshly-built wheel instead of the PyPI one."
        )

    # 统一处理 fork 和非 fork PR 的输入字典，简化代码结构
    inputs = {
        "target_stage": stage_name,
        "pr_head_sha": pr_head_sha if is_fork else None, # 初始设置，后续可能调整
    }

    # 对于非 fork PRs，直接使用分支引用并重置输入
    if not is_fork:
        ref = pr.head.ref
        inputs = {"target_stage": stage_name} # 重置为仅包含 target_stage

    # 如果检测到内核变更且目标阶段不是 AMD（AMD 有独立管道），则设置 include_wheel_build
    if kernel_changes and not is_amd_stage:
        inputs["include_wheel_build"] = "true"
        # 确保 pr_head_sha 被设置以支持 filter-api 检测内核变更
        if not is_fork:
            inputs["pr_head_sha"] = pr.head.sha

    # 记录时间并触发工作流
    dispatch_time = time.time()
    # ... 后续触发逻辑 ...
```

.github/workflows/pr-test.yml

定义 CI 工作流的主要配置文件，添加新输入并调整逻辑以支持内核构建在 target_stage 模式下运行，影响整个测试流程。

```
# 在工作流的 on.workflow_dispatch.inputs 部分添加新输入
on:
  workflow_dispatch:
    inputs:
      include_wheel_build:
        description: "When set with target_stage, also run sgl-kernel-build-wheels so the target stage uses the freshly-built kernel (for /rerun-stage on PRs that modify sgl-kernel/)"
        required: false
        type: boolean
        default: false
      # 其他现有输入 ...

# 在 jobs.check-changes.outputs 中调整 sgl_kernel 输出逻辑
jobs:
  check-changes:
    outputs:
      sgl_kernel: ${{ (!inputs.target_stage || inputs.include_wheel_build) && (steps.filter-api.outputs.sgl_kernel || steps.filter.outputs.sgl_kernel) }}
      # 说明：当 target_stage 未设置或 include_wheel_build 为 true 时，保留真实检测值；否则强制为 false，避免构建跳过

  # 在 sgl-kernel-build-wheels 作业中更新条件以允许 target_stage 模式下运行
  sgl-kernel-build-wheels:
    if: always() && (needs.call-gate.result == 'success' || (inputs.target_stage && inputs.include_wheel_build))
    # 说明：使用 always() 确保作业可运行，并结合 call-gate 结果或 include_wheel_build 标志决定

  # 更新 validate-target-stage 步骤，仅在 include_wheel_build 未设置时失败
  validate-target-stage:
    if: inputs.target_stage && !inputs.include_wheel_build && (steps.filter-api.outputs.sgl_kernel == 'true' || steps.filter.outputs.sgl_kernel == 'true')
    run: |
      echo "::error::Cannot use /rerun-stage when PR has sgl-kernel changes. The handler should have auto-set include_wheel_build=true; if you see this, the handler may be out-of-date."
      exit 1
    # 说明：提供安全网，防止内核变更时错误使用旧 wheel
```

评论区精华

由于没有 review 评论，讨论亮点为空。PR 由作者直接合并，表明变更经过充分测试且无争议。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险在于 CI 逻辑的复杂性：如果 include_wheel_build 标志设置错误（例如，检测逻辑误判或条件不匹配），可能导致 sgl-kernel-build-wheels 不必要地运行（增加 CI 时间和资源）或错误跳过（导致测试使用旧 wheel）。此外，新输入的添加可能与其他工作流调用者不兼容，但默认值为 false 确保了向后兼容。验证步骤 validate-target-stage 的调整提供了安全网，在标志缺失时会快速失败并给出清晰错误信息，降低了静默失败的风险。
- 影响：此变更直接影响 CI 流程，确保修改 sgl-kernel/ 的 PR 在 /rerun-stage 命令下能正确测试内核变更，提高了测试准确性和开发效率。对于开发者，这减少了调试时的误导；对于系统，增加了 CI 的健壮性，但可能略微增加构建时间当内核变更时。影响范围限于 CI 基础设施，对用户功能无直接影响，且默认行为保持不变，确保现有工作流不受影响。
- 风险标记：CI 流程变更，配置逻辑风险

关联脉络

- PR #23008 ci: use rerun_failed_jobs for skipped workflows in /rerun-failed-ci: 都涉及 CI 重新运行命令的改进，共享类似的脚本处理逻辑和基础设施变更背景。