PR #23008 完整报告

sgl-project/sglang

ci: use rerun_failed_jobs for skipped workflows in /rerun-failed-ci

合并时间：2026-04-22 14:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23008

执行摘要

- 一句话：修复 /rerun-failed-ci 命令对跳过工作流的处理，避免不必要的完整 CI 重启。
- 推荐动作：值得 CI 维护者和基础设施工程师精读，关注 sgl-kernel 轮子构建检查的实现细节和跳过工作流的处理策略，以理解 CI 自动化优化方向。

功能与动机

根据 PR body，问题在于当工作流结论为 'skipped'（例如来自级联快速失败）时，处理程序调用 run.rerun() 重启整个工作流，导致 /rerun-failed-ci 行为类似于完整 CI 重启，而非仅重试失败作业。

实现拆解

1. 检查 sgl-kernel 轮子构建状态：在 scripts/ci/utils/slash_command_handler.py 的 handle_rerun_failed_ci 函数中，更新逻辑以检查所有轮子构建（CUDA 和 ARM）是否成功，使用检查运行的显示名称 "Build Wheel" 和 "Build Wheel Arm"，确保依赖作业不会因缺失构件而失败。
2. 处理跳过工作流：对于结论为 "skipped" 的工作流，使用 rerun_failed_jobs() 方法，如果没有失败或取消的作业，则回退到 run.rerun()，避免重启已通过的作业。
3. 集成到现有命令处理：修改 handle_rerun_failed_ci 函数的控制流，调整异常处理和打印日志，以匹配新逻辑。
4. 测试配套：无显式测试变更，但 PR body 提到测试计划，需验证仅失败作业被重新运行。

关键文件：

- scripts/ci/utils/slash_command_handler.py（模块 CI 工具；类别 infra；类型 infrastructure；符号 handle_rerun_failed_ci）：这是 /rerun-failed-ci 命令的处理程序，核心变更在此文件中，涉及控制流调整和外部 API 调用优化。

关键符号：handle_rerun_failed_ci

关键源码片段

scripts/ci/utils/slash_command_handler.py

这是 /rerun-failed-ci 命令的处理程序，核心变更在此文件中，涉及控制流调整和外部 API 调用优化。

```
def handle_rerun_failed_ci(gh_repo, pr, comment, user_perms, react_on_success=True):
    # ... 现有代码 ...
    head_sha = pr.head.sha
    print(f"Checking workflows for commit: {head_sha}")

    # 检查 sgl-kernel 轮子构建状态：确保所有轮子（CUDA 和
    ARM）都已成功构建，避免依赖作业因缺失构件而失败
    kernel_wheel_built = False
    if sgl_kernel_changes:
        try:
            wheel_builds = [
                cr for cr in gh_repo.get_commit(head_sha).get_check_runs()
                if cr.name.startswith("Build Wheel") # 匹配显示名称，而非 YAML job id
            ]
            kernel_wheel_built = bool(wheel_builds) and all(
                cr.conclusion == "success" for cr in wheel_builds
            )
            print(
                f"All {len(wheel_builds)} kernel wheel build(s) passed - using rerun_failed_jobs"
                if kernel_wheel_built
                else f"Kernel wheel not fully built "
                    f"({sum(1 for c in wheel_builds if c.conclusion == 'success')}"
                    f"/{len(wheel_builds)} success) - will use full rerun"
            )
        except Exception as e:
            print(f"Failed to check kernel wheel status: {e} - falling back to full rerun")

    # 处理工作流运行：针对失败或跳过的工作流，优先使用 rerun_failed_jobs() 仅重试失败作业
    runs = gh_repo.get_workflow_runs(head_sha=head_sha)
    rerun_count = 0
    for run in runs:
        if run.conclusion in ["failure", "skipped"]:
            if run.conclusion == "skipped" and not kernel_wheel_built:
                # 如果整个工作流被跳过且轮子未构建，使用完整重新运行，因为无作业可针对
                run.rerun()
            else:
                # 使用 GitHub API 的 rerun_failed_jobs() 方法，仅重新运行失败作业及其依赖作业
                run.rerun_failed_jobs()
            rerun_count += 1
    # ... 后续代码，如打印结果和返回状态 ...
```

评论区精华

无 review 评论，但提交历史显示多次迭代：初始实现后修复轮子构建检查匹配问题、简化跳过工作流处理逻辑、最终恢复结论为 "skipped" 工作流的完整重新运行分支，以确保正确处理无作业运行的情况。

- 暂无高价值评论线程

风险与影响

- 风险： 1) 回归风险：控制流变更可能影响其他 CI 场景，如处理不同工作流状态时逻辑错误； 2) 依赖外部 API：GitHub API 的 rerun_failed_jobs() 行为变化或检查运行名称变更可能导致意外失败； 3) 兼容性：sgl-kernel 轮子构建检查逻辑变更可能遗漏边缘情况，如新构建作业添加时未更新检查。
  - 影响：对用户影响：提高 CI 重试效率，减少不必要的作业运行时间，加速反馈循环；对系统影响：优化资源使用，降低 CI 负载，但需确保新逻辑正确处理所有工作流状态；对团队影响：简化 CI 维护，减少手动干预，但需监控变更效果以防引入新问题。
  - 风险标记：控制流变更，外部 API 依赖，缺少测试覆盖

关联脉络

- PR #22828 ci: enable /rerun-test for multimodal gen PR tests: 同样涉及 CI 命令处理扩展，扩展 /rerun-test 命令功能。
- PR #23420 Update pr-test-xeon.yml cancel-in-progress config: 涉及 CI 配置优化，调整并发策略以加速反馈。
- PR #23417 [ci] split stage-c-test-4-gpu-b200 to enable a low-disk runner pool: 涉及 CI 测试套件分割，优化资源分配，与本 PR 同属基础设施改进。