PR #1961 完整报告

radixark/miles

docker: keep the cu12 dependency markers after checking out sglang-miles

合并时间：2026-07-30 08:16

原文链接：http://prhub.com.cn/radixark/miles/pull/1961

执行摘要

- 一句话：修复 CUDA12 镜像 sglang 脏树阻断发布构建
- 推荐动作：值得快速阅读，作为一个典型的“上游镜像脏树 + 依赖标记污染”CI 问题的修复案例。其价值不在于代码量，而在于对根因的完整定位：作者先确认是上游镜像构建行为导致，再论证“只用 checkout -f 不够”，必须重放 sed 改写才能避免可编辑安装元数据泄漏 cu13 标记。可关注的点是 sed 模式与上游文件耦合带来的维护成本，后续升级 sglang-miles 时需要注意同步校验。

功能与动机

PR body 明确指出主分支的 Docker Build & Push 在 Install sglang-miles 阶段失败：git checkout 因 python/pyproject.toml 的本地改动而中止。原因在于 sglang 官方 v0.5.16-cu129 镜像在构建时通过 sed 改写了依赖标记但不提交，而 v0.5.15-cu129 是干净的，因此这是 v0.5.16 CUDA 12 镜像的新问题而非仓库自身的回归。Issue #1836 也佐证了同一机制：cu13 标记泄漏到 cu12 镜像会导致依赖解析拉错包（cuDNN 被降级）。

实现拆解

1. 强制 checkout：在 docker/Dockerfile 的 sglang 安装阶段，将 git checkout ${SGLANG_COMMIT} 和 git checkout FETCH_HEAD 都改为 git checkout -f，绕过脏工作树导致的 abort。
2. 按条件重放依赖改写：在 checkout 后新增 if ["${ENABLE_CUDA_13}" != "1"] 分支，对 python/pyproject.toml 执行与 sglang 上游 Dockerfile 完全一致的三个 sed：cuda-python>=13.0 → cuda-python>=12,<13，flashinfer_python[cu13] → flashinfer_python[cu12]，nvidia-cutlass-dsl[cu13] → nvidia-cutlass-dsl。这样保证随后 pip install -e "python[all]" --no-deps 发布到环境中的依赖元数据与基础镜像一致。
3. 不做额外处理：humming-kernels[cu13] 刻意不动，与上游行为保持一致；sglang-miles 分支相对 v0.5.16 只多了 xxhash 一项，改写后树的依赖标记与基础镜像一致。
4. 测试配套：本 PR 未新增自动化测试，作者在真实 lmsysorg/sglang:v0.5.16-cu129 镜像上复现失败并验证修复序列有效。

关键文件：

- docker/Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure）：唯一的变更文件，修复 sglang-miles checkout 时因上游镜像脏工作树导致的构建失败，并重放 cu12 依赖

标记改写，是整个 PR 的核心。

关键符号：未识别

关键源码片段

docker/Dockerfile

唯一的变更文件，修复 sglang-miles checkout 时因上游镜像脏工作树导致的构建失败，并重放 cu12 依赖标记改写，是整个 PR 的核心。

```
# 安装 sglang-miles：先强制 checkout，再按需重写 cu12 依赖标记。
# CUDA 12 镜像会在构建时用 sed 改写 pyproject.toml 里的 cu13 标记，
# 且从不提交，普通 checkout 会因"本地改动"中止。
RUN cd /sgl-workspace/sglang && \
    git fetch origin ${SGLANG_BRANCH} && \
    if [ -n "${SGLANG_COMMIT}" ]; then \
        git checkout -f ${SGLANG_COMMIT}; \
    else \
        git checkout -f FETCH_HEAD; \
    fi && \
    # 仅 cu12 镜像需要把标记从 cu13 改回 cu12，
    # 否则 sglang 的可编辑安装元数据会携带 cu13 依赖，
    # 后续 pip 解析会在 cu12 环境里拉错包。
    if [ "${ENABLE_CUDA_13}" != "1" ]; then \
        sed -i 's/cuda-python>=13\.0/cuda-python>=12,<13/' python/pyproject.toml && \
        sed -i 's/flashinfer_python\[cu13\]/flashinfer_python[cu12]/' python/pyproject.toml && \
        sed -i 's/nvidia-cutlass-dsl\[cu13\]/nvidia-cutlass-dsl/' python/pyproject.toml; \
    fi && \
    pip install -e "python[all]" --no-deps
```

评论区精华

review 评论很少：guapisolo 仅回复 approved to unblock.，没有展开技术讨论。最有价值的验证来自作者自己在 issue 评论中的补充：他在真实 v0.5.16-cu129 镜像上确认脏文件恰好只有 python/pyproject.toml 的三个标记改动，复现失败信息与 PR body 一致，并验证新逻辑在 ENABLE_CUDA_13=0 下能完成 checkout + sed 改写。

- 对真实 v0.5.16-cu129 镜像的验证 (testing): 验证通过，修复序列在真实镜像上有效。
- Review 批准与解除阻塞 (other): 已批准并合并。

风险与影响

- 风险：风险集中在 sed 模式对上游文件结构的强依赖：sglang-miles 分支后续如果调整这三行依赖的写法，sed 可能静默失败，cu12 镜像上会残留 cu13 标记，重新触发与 #1836 相同的依赖污染（pip 解析拉取 cu13 包）。强制 checkout 理论上会丢弃上游未提交的改动，但在该受控构建场景下不构成实际风险。改动无自动化测试覆盖，依赖人工在发布路径上验证。整体影响面仅限 Docker 发布构建，不触及运行时逻辑。

- 影响：直接解除 v0.5.16-cu129 发布构建的阻塞；PR CI 使用默认 v0.5.16（cu130）镜像，不经过 sed 改写路径，故不受影响。对用户无感知，对系统影响局限在镜像构建阶段的依赖正确性，避免 cu13 依赖被错误安装到 cu12 镜像上。对团队而言，这是一个小而关键的 CI 修复，恢复了发布流水线可用性。
- 风险标记：依赖上游文件结构，发布构建路径，无测试覆盖

关联脉络

- PR #1836 ci: restore the image's cuDNN pin after reconciling requirements: 同一类问题的另一表现：cu13 标记泄漏到 cu12 镜像导致 pip 解析拉错依赖（cuDNN 被降级）。本 PR 的 sed 重写正是从根源上避免 cu13 标记进入可编辑安装元数据，与 #1836 的恢复逻辑互补。