PR #29986 完整报告

sgl-project/sglang

[AMD]: hot-patch transformers dynamic_module_utils symlink bug

合并时间：2026-07-03 18:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29986

执行摘要

- 一句话：热补丁修复 AMD 镜像 transformers 符号链接 bug
- 推荐动作：该 PR 是典型的临时热修复，适合快速合入以解除 AMD 部署阻塞。建议精读 hot-patch 的实现方式，特别是其自跳过和断言保护模式，可作为同类紧急修复的参考模板。

功能与动机

transformers 5.12.1 的 _compute_local_source_files_hash 对自定义代码模块文件调用 .resolve()，在 HuggingFace 缓存中 snapshot 文件是到 blobs 的相对符号链接，解析后 get_relative_import_files 无法找到兄弟模块，导致 Kimi-K2.6 等依赖相对导入的模型在处理器初始化时抛出 FileNotFoundError。这阻塞了 ROCm/MI35x 镜像上的 Kimi K2.6 分离部署运行。

实现拆解

1. 定位问题：在 docker/rocm.Dockerfile 中分析得知 transformers 5.12.1 的 _compute_local_source_files_hash 对自定义代码文件调用 .resolve()，导致在 HF 缓存符号链接布局下丢失相对导入上下文。
2. 应用热补丁：在 Dockerfile 末尾添加一个 RUN 步骤，使用 inline Python 脚本读取并修改 transformers/dynamic_module_utils.py，将两处 .replace("Path(resolved_module_file).resolve()", "Path(resolved_module_file)") 和 .replace("Path(source_file).resolve()", "Path(source_file)") 替换，去掉 .resolve() 调用。
3. 健壮性检查：脚本先检查目标字符串是否存在：若不存在则认为上游已修复并跳过；若存在但替换后内容无变化则断言失败，阻止静默空操作。
4. 仅限 ROCm 镜像：变更仅位于 docker/rocm.Dockerfile，NV/CUDA 路径不受影响；补丁会在上游 transformers 发布正式修复后自动跳过。

关键文件：

- docker/rocm.Dockerfile（模块 部署脚本；类别 infra；类型 infrastructure）：唯一变更文件，添加热补丁步骤解决 transformers 符号链接问题，直接影响 AMD 镜像构建。

关键符号：未识别

关键源码片段

docker/rocm.Dockerfile

唯一变更文件，添加热补丁步骤解决 transformers 符号链接问题，直接影响 AMD 镜像构建。

```
# 热补丁：transformers dynamic_module_utils 符号链接 bug（v5.12.1）
# _compute_local_source_files_hash 对自定义代码模块文件调用 .resolve()，
# 在 HF 缓存 snapshots/<hash>/x.py -> blobs/<blob> 符号链接下，
# get_relative_import_files 会到 blobs 目录寻找兄弟模块而失败。
# 此补丁去掉 .resolve() 以保持 snapshot 路径，镜像上游 transformers PR #46618。
RUN python3 - <<'PY'
import pathlib
import transformers.dynamic_module_utils as m

MARKS = ["Path(resolved_module_file).resolve()", "Path(source_file).resolve()"]
path = pathlib.Path(m.__file__)
src = path.read_text()
if not any(mark in src for mark in MARKS):
    # 上游已修复，跳过补丁
    print("transformers dynamic_module_utils already fixed; no patch needed")
else:
    # 替换两处 .resolve() 调用为直接 Path 构造
    patched = (
        src.replace("Path(resolved_module_file).resolve()", "Path(resolved_module_file)")
        .replace("Path(source_file).resolve()", "Path(source_file)")
    )
    # 断言确保实际发生了替换，防止静默空操作
    assert patched != src, "FATAL: transformers symlink patch matched nothing"
    path.write_text(patched)
    print("patched transformers dynamic_module_utils.py (symlink hash fix)")
PY
```

评论区精华

审查者 HaiShaw 要求合并前进行构建和测试验证，贡献者 bingxche 确认构建镜像成功（GitHub Actions run #28633959331），并使用测试镜像运行 Kimi-K2.6 DI 测试通过（run #28645858783），随后合并。无其他技术争议或未解决疑虑。

- 构建和测试验证 (testing): 验证通过，可以合并。

风险与影响

- 风险：低风险。变更仅作用于 AMD ROCm Docker 镜像构建阶段，且包含主动跳过和断言保护，避免对已修复版本误操作。但需注意：若未来 transformers 代码结构变动导致目标字符串变化，补丁可能无法应用，但断言会提前失败，不会静默破坏构建。对通用代码路径无影响。
- 影响：直接影响：修复 AMD ROCm 镜像上使用相对导入的 trust_remote_code 模型（如 Kimi-K2.6）的部署问题。间接影响：无，因为仅修改 Dockerfile 构建步骤，不影响运行时逻辑、性能或兼容性。团队受益于修复了 AMD 平台上的关键阻塞问题。

- 风险标记：构建步骤变更，依赖上游库未发布补丁

关联脉络

- 暂无明显关联 PR