# PR #2039 完整报告

radixark/miles

[AMD] fix mooncake object store support

合并时间：2026-08-01 06:09

原文链接：http://prhub.com.cn/radixark/miles/pull/2039

## 执行摘要

- 一句话：修复 ROCm 上 mooncake 对象存储 Python 层缺失
- 推荐动作：该 PR 值得快速浏览，因为它揭示了 ROCm 基础镜像从源码构建扩展时常见的 Python 层缺失问题，并给出了一个轻量级修复模式。建议关注 Dockerfile.rocm 中 cp -n 的用法，以及基础镜像版本与 wheel 版本的对应关系；后续升级 mooncake 版本时需要保持两者同步。

## 功能与动机

在 ROCm 平台上，mooncake.structured_object_store 模块缺失，导致 MooncakeObjectStore 在 actor 启动时构造失败，破坏了三个使用 mooncake 对象存储后端的 ROCm CI 测试。PR body 指出，SGLang ROCm 基础镜像从源码构建 Mooncake 时，CMake install 规则按显式文件名安装 Python 文件，遗漏了 structured_object_store.py 等纯 Python 文件；而 CUDA 基础镜像通过 pip 安装则不受影响。

## 实现拆解

1. 升级 ROCm 基础镜像版本：在 docker/Dockerfile.rocm 和 docker/build.py 中，将 rocm720-mi35x 的 SGLANG_IMAGE_TAG 从 v0.5.14-rocm720-mi35x-20260627 更新为 v0.5.16-rocm720-mi35x-20260730。旧基础镜像的本地扩展缺少 setup(config: dict) 和 remove(key, force) API，新版本补齐了这些原生 API。
2. 从 wheel 补齐 Python 层：在 docker/Dockerfile.rocm 中新增 ARG MOONCAKE_VERSION=0.3.12.post1，并通过 pip install --no-deps --target=/tmp/mc 安装对应版本的 mooncake-transfer-engine-non-cuda wheel，然后使用 cp -n 将 /tmp/mc/mooncake/*.py 拷贝到已安装的 mooncake 包目录。cp -n 只填充缺失文件，不覆盖 HIP 链接的原生模块，保证原生构建不受影响。
3. 验证：PR body 报告了 MI350 Miles 测试矩阵（2/4/8 GPU）共 14 个测试全部通过，无跳过，覆盖了 mooncake 相关的 test_qwen3_30B_A3B/test_baseline.py、test_qwen2.5_0.5B_gsm8k_async_short.py 和 test_qwen2.5_0.5B_gsm8k_short.py。

关键文件：

- docker/Dockerfile.rocm（模块 部署脚本；类别 infra；类型 core-logic；符号 MOONCAKE_VERSION）：核心修复文件：升级 ROCm 基础镜像并从 wheel 补齐 mooncake 纯 Python 文件，解决 structured_object_store 模块缺失问题。

- docker/build.py（模块 部署脚本；类别 infra；类型 configuration）：构建配置中同步更新了 rocm720-mi35x 的镜像标签，与 Dockerfile.rocm 保持一致。

关键符号：未识别

## 关键源码片段

### docker/Dockerfile.rocm

核心修复文件：升级 ROCm 基础镜像并从 wheel 补齐 mooncake 纯 Python 文件，解决 structured_object_store 模块缺失问题。

```
# docker/Dockerfile.rocm（关键片段）
# 基础镜像从 v0.5.14 升级到 v0.5.16，后者提供 mooncake 原生 setup/remove API
ARG SGLANG_IMAGE_REPO=rocm/sgl-dev
ARG SGLANG_IMAGE_TAG=v0.5.16-rocm720-mi35x-20260730

# 固定与基础镜像匹配的 mooncake wheel 版本
ARG MOONCAKE_VERSION=0.3.12.post1

# make install 只安装原生模块，漏掉纯 Python 文件（如 structured_object_store.py）
# 这里从发布 wheel 中补齐缺失文件：
# --no-deps 避免带入无关依赖；--target=/tmp/mc 解包到临时目录
RUN pip install --no-deps --target=/tmp/mc \
    "mooncake-transfer-engine-non-cuda==${MOONCAKE_VERSION}" && \
    # cp -n 只填充缺失文件，不覆盖 HIP 链接的原生 .so 模块
    cp -n /tmp/mc/mooncake/*.py \
        "$(python3 -c 'import mooncake, os; print(os.path.dirname(mooncake.__file__))')" && \
    # 清理临时目录，保持镜像精简
    rm -rf /tmp/mc
```

## 评论区精华

Review 由 guapisolo 批准，仅有一条评论为 gemini-code-assist[bot] 的自动化停用提示，无实质性讨论。PR body 中对比了新旧基础镜像的能力表，说明仅升级基础镜像不足以解决问题，必须同时补齐 Python 文件。

- gemini-code-assist 停用提示 (other): 无实际技术内容，不影响 PR。
- 基础镜像升级的必要性 (design): 通过组合方案（升级基础镜像 + cp -n 补齐 Python 文件）解决。

## 风险与影响

- 风险：
  1. 原生模块与 wheel Python 文件版本兼容性：使用 pip install --no-deps 安装 wheel 后 cp -n 拷贝 Python 文件，若 wheel 版本与基础镜像中已编译原生模块版本不完全匹配，可能导致顶层 Python API 与底层原生符号不一致。当前固定 MOONCAKE_VERSION=0.3.12.post1，但未来升级基础镜像时需要同步核对。

2. cp -n 的静默跳过：cp -n 会跳过已存在的文件，若基础镜像中的 Python 文件版本过旧，不会得到更新，可能遗留隐患。当前场景下该行为是有意为之（避免覆盖 HIP 链接模块），但需要注释说明。
3. 临时目录清理：拷贝后执行 rm -rf /tmp/mc，若中途失败可能导致镜像残留，但影响有限。
4. 仅影响 ROCm 路径：CUDA 基础镜像不受影响，但本变更仅覆盖 docker 镜像构建，若用户自建镜像需自行应用类似修复。 - 影响：影响范围集中在 ROCm 平台的 docker 镜像构建路径，主要收益是修复 mooncake 对象存储在 ROCm 上的可用性，解锁相关 CI 测试（如 test_qwen3_30B_A3B/test_baseline.py、test_qwen2.5_0.5B_gsm8k_async_short.py、test_qwen2.5_0.5B_gsm8k_short.py）。对 CUDA 用户无影响。团队维护成本增加一个版本 ARG 和构建步骤，但复杂度低。 - 风险标记：平台特定修复，版本耦合风险，镜像构建路径

## 关联脉络

- PR #1961 docker: keep the cu12 dependency markers after checking out sglang-miles: 同为 docker 构建相关修复，处理基础镜像与依赖安装问题。
- PR #1795 Bump sglang to v0.5.16: 涉及基础镜像升级和依赖栈适配，与本次升级 ROCm 基础镜像到 v0.5.16 相关。
- PR #2040 [AMD] DeepSeek-V4: use the ROCm precision-parity norm path: 同为 AMD/ROCm 平台支持相关 PR，属于同一平台功能演进脉络。