PR #23073 完整报告

sgl-project/sglang

[AMD] mirror nightly images to local registry and prefer LAN pulls

合并时间：2026-04-17 19:49

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23073

执行摘要

- 一句话：为 AMD CI 新增内网 Docker registry 镜像和 LAN 优先拉取策略，解决 Docker Hub rate limit 问题。
- 推荐动作：该 PR 值得运维和 CI 工程师精读，重点关注镜像拉取策略的设计决策（如优先级顺序和重试逻辑），以及硬编码 IP 和代码重复等可维护性改进点。

功能与动机

根据 PR body 描述，主要动机是“preventing docker rate limit issue on mi300x”，即避免在 AMD MI300x CI 环境中因频繁拉取 Docker Hub 镜像而触发的速率限制，通过镜像到内网 registry 和优先 LAN 拉取来提升 CI 的可靠性和效率。

实现拆解

1. 扩展 nightly 发布工作流以镜像镜像到内网 registry：修改 .github/workflows/release-docker-amd-nightly.yml 和 release-docker-amd-rocm720-nightly.yml，在 publish job 成功后新增 Save published image tag 步骤持久化镜像标签为短时 artifact，并添加 push_local_registry job 从 artifact 读取标签，将 rocm/sgl-dev 镜像重新打标并推送到内网 registry 172.29.8.23:5000。
2. 更新 AMD CI 启动脚本以优化镜像拉取策略：修改 scripts/ci/amd/amd_ci_start_container.sh 和 amd_ci_start_container_disagg.sh，新增 LOCAL_DOCKER_REGISTRY 变量定义内网 registry 地址，并重构 find_latest_image 函数，将镜像查找顺序改为：先检查本地 runner 缓存，再尝试内网 registry（使用 docker manifest inspect 验证），最后回退到公共 Docker Hub 拉取。
3. 为公共 Docker Hub 拉取添加重试逻辑：在 CI 启动脚本中新增 retry_with_backoff 函数，实现指数退避重试，仅用于 Docker Hub 登录和拉取操作，以减少因网络问题导致的失败。
4. 配套调整与注意事项：无测试或文档配套改动，但 PR 中提及暂时禁用 docker cache（通过一个提交），这可能影响性能但未详细说明原因。

关键文件：

- .github/workflows/release-docker-amd-nightly.yml（模块 工作流配置；类别 infra；类型 infrastructure）：这是核心工作流文件，定义了 nightly 镜像发布到 Docker Hub 后镜像到内网 registry 的流程，直接影响镜像的可用性和 CI 依赖。

- scripts/ci/amd/amd_ci_start_container.sh（模块 CI 脚本；类别 infra；类型 infrastructure；符号 find_latest_image, retry_with_backoff）：关键 CI 启动脚本，实现了镜像拉取优先级策略和重试逻辑，直接决定 CI 运行时的镜像来源。
- .github/workflows/release-docker-amd-rocm720-nightly.yml（模块 工作流配置；类别 infra；类型 infrastructure）：类似标准 AMD nightly 工作流，扩展了 ROCm 7.2 版本的镜像镜像流程，确保不同 ROCm 版本的镜像同步。
- scripts/ci/amd/amd_ci_start_container_disagg.sh（模块 CI 脚本；类别 infra；类型 infrastructure；符号 find_latest_image, retry_with_backoff）：解聚模式下的 CI 启动脚本，与标准脚本类似，实现了相同的镜像拉取优化，确保一致性。

关键符号：find_latest_image, retry_with_backoff

关键源码片段

scripts/ci/amd/amd_ci_start_container.sh

关键 CI 启动脚本，实现了镜像拉取优先级策略和重试逻辑，直接决定 CI 运行时的镜像来源。

```
# 定义内网 registry 地址（硬编码，建议改为环境变量以提升可维护性）
LOCAL_DOCKER_REGISTRY="172.29.8.23:5000"

# 指数退避重试函数，仅用于 Docker Hub 登录和拉取，减少网络问题影响
retry_with_backoff() {
    local max_attempts=$1; shift
    local attempt=1
    local wait_secs=30
    local jitter=$(( RANDOM % 30 )) # 添加随机抖动，避免并发作业同时重试
    while true; do
        if "$@"; then
            return 0 # 命令成功则退出
        fi
        if (( attempt >= max_attempts )); then
            echo "Error: '$*' failed after ${max_attempts} attempts" >&2
            return 1 # 达到最大尝试次数后失败
        fi
        local sleep_time=$(( wait_secs + jitter ))
        echo "Attempt ${attempt}/${max_attempts} failed. Retrying in ${sleep_time}s…" >&2
        sleep "${sleep_time}"
        (( attempt++ ))
        (( wait_secs = wait_secs * 2 > 300 ? 300 : wait_secs * 2 )) # 指数退避，上限 300 秒
        jitter=$(( RANDOM % 30 ))
    done
}

# 更新 find_latest_image 函数，优化镜像查找优先级
find_latest_image() {
    local gpu_arch=$1
    local base_tag days_back image_tag image_id remote_tags
    case "${gpu_arch}" in
```

```
        mi30x) base_tag="${MI30X_BASE_TAG}" ;;
        mi35x) base_tag="${MI35X_BASE_TAG}" ;;
        *) echo "Error: unsupported GPU architecture '${gpu_arch}'" >&2; return 1 ;;
    esac
    # 第一步：检查本地 runner 缓存
    for days_back in {0..6}; do
        image_tag="${base_tag}-$(date -d "${days_back} days ago" +%Y%m%d)"
        image_id=$(docker images -q "rocm/sgl-dev:${image_tag}")
        if [[ -n "$image_id" ]]; then
            echo "Found cached image locally: rocm/sgl-dev:${image_tag}" >&2
            echo "rocm/sgl-dev:${image_tag}"
            return 0
        fi
    done
    # 第二步：尝试内网 registry
    for days_back in {0..6}; do
        image_tag="${base_tag}-$(date -d "${days_back} days ago" +%Y%m%d)"
        if docker manifest inspect --insecure "${LOCAL_DOCKER_REGISTRY}/rocm/sgl-dev:
        ${image_tag}" >/dev/null 2>&1; then
            echo "Found available image: ${LOCAL_DOCKER_REGISTRY}/rocm/sgl-dev:${image_tag}
            " >&2
            echo "${LOCAL_DOCKER_REGISTRY}/rocm/sgl-dev:${image_tag}"
            return 0
        fi
    done
    # 第三步：回退到公共 Docker Hub 拉取
    # ... 后续逻辑省略
}
```

评论区精华

review 评论中，gemini-code-assist[bot] 指出两个主要问题：

- 硬编码内网 registry IP 地址：在 amd_ci_start_container.sh 和 amd_ci_start_container_disagg.sh 中直接设置 LOCAL_DOCKER_REGISTRY="172.29.8.23:5000"，建议改为环境变量以提升可维护性。
- 重复的重试函数：retry_with_backoff 函数在两个脚本中重复定义，建议抽取到共享脚本中遵循 DRY 原则。 这些讨论未在 PR 中得到解决，但 PR 已合并，可能作为未来改进点。
- 硬编码内网 registry IP 地址的可维护性问题 (design): 建议未在 PR 中采纳，但问题被记录为未来改进点。
- 重复重试函数的代码重复问题 (design): 建议未在 PR 中实现，可能作为技术债务留待后续重构。

风险与影响

- 风险：技术风险包括：

- 硬编码配置风险：内网 registry IP 地址硬编码在脚本中，如果基础设施变更（如 registry 地址更新），需要手动修改多个文件，可能导致 CI 失败。
- 代码重复风险：retry_with_backoff 函数在两个 CI 脚本中重复，增加维护成本和潜在不一致性。
- 依赖外部服务：内网 registry 的可用性依赖内部网络，如果 registry 服务中断，CI 可能回退到 Docker Hub 但仍有速率限制风险。
- 缺少测试覆盖：变更主要涉及基础设施脚本，没有添加单元测试，回归风险较低但依赖手动验证。
- 影响：影响范围主要集中在 AMD CI 管道和依赖 nightly 镜像的测试环境：
- 对用户影响：AMD CI 用户（如开发者、测试人员）将受益于更可靠的镜像拉取，减少因 Docker Hub rate limit 导致的 CI 失败，提升开发效率。
- 对系统影响：CI 管道拉取镜像时优先使用内网 LAN，降低外部依赖和网络延迟，可能加快 CI 执行速度；但内网 registry 需要额外维护和存储资源。
- 对团队影响：简化了 CI 配置管理，但引入了新的基础设施组件（内网 registry），需要团队确保其高可用性。
- 风险标记：硬编码配置，代码重复

关联脉络

- PR #22974 [AMD] fix AMD CI gate: 涉及 AMD CI 修复，与本 PR 同属 AMD CI 基础设施改进，共享 run-ci 标签。
- PR #22952 [AMD] Add SGLANG_MORI_MOE_MAX_INPUT_TOKENS to truncate dispatch before MoE.: 同样针对 AMD 平台，涉及性能优化和文档更新，展示团队对 AMD 生态的持续投入。
- PR #22274 [AMD] CI Job Monitor: fix queue time, utilization, and summary metrics: 修复 AMD CI 监控脚本，与本 PR 在 CI 基础设施层面相关，强调 AMD 环境下的运维改进。