PR #2131 完整报告

radixark/miles

ci(docker): rebuild scheduled images at least once every 24h

合并时间：2026-08-04 06:12

原文链接：http://prhub.com.cn/radixark/miles/pull/2131

执行摘要

- 一句话：Docker 定时构建增加 24h 陈旧回退
- 推荐动作：这是一个小而清晰的 CI 基础设施改进，适合快速通读以了解构建触发策略的权衡。值得关注的设计点包括：用缓存文件行号记录时间戳、仅在构建触发时写回以保证时间戳语义、旧缓存自动迁移。若后续需要调整重建周期，只需修改 MAX_BUILD_AGE_SECONDS 一处。

功能与动机

定时构建仅在 sglang / Megatron-LM / miles-wheels 上游移动时触发，而 miles 仓库自身的提交被刻意排除在触发条件之外（避免重建过于频繁）。PR body 明确指出："When upstream is quiet the image is never rebuilt, so radixark/miles:dev silently drifts behind the miles repo itself"，因此需要加入陈旧上限，确保镜像最多落后一天。

实现拆解

1. 读取时间戳：在 .github/workflows/docker-build.yml 的 check-upstream 作业中，从缓存文件追加读取第 6 行 LAST_BUILD_EPOCH；缓存文件不存在时初始化为空字符串。
2. 新增陈旧回退：在原有上游 SHA 比较逻辑之后，若 SHOULD_BUILD=false，则检查时间戳：无效（旧 5 行缓存或乱码）时强制重建一次（完成迁移）；有效且距今 ≥ 24h 时也强制重建，并输出对应日志。
3. 写回时间戳：保存缓存时追加第 6 行 NOW_EPOCH。由于缓存只在真正触发构建时写回，该时间戳天然等价于 " 最后一次构建决策 " 的时刻，无需额外 API 调用或新状态。
4. 同步文档：更新 docs/ci/02-docker-build.md，在 check-upstream 段落补充 24h 陈旧界限说明，并将触发器表中 schedule 行的行为改为 "build if upstream moved or last build ≥ 24h ago"。

补充说明：手动 workflow_dispatch 或 push-to-main 触发的构建不触碰该缓存，因此不会刷新时间戳；最坏情况下，一次手动构建后，下一次计划运行仍可能因陈旧而重建，这符合设计预期。

关键文件：

- .github/workflows/docker-build.yml（模块 构建工作流；类别 infra；类型 infrastructure）：核心改动文件：为 check-upstream 增加 24h 陈旧回退逻辑，读取并写入缓存第 6 行时间戳。

- docs/ci/02-docker-build.md（模块 文档；类别 docs；类型 documentation）：同步更新 CI 文档，说明 24h 陈旧重建规则和触发器表。

关键符号：未识别

关键源码片段

.github/workflows/docker-build.yml

核心改动文件：为 check-upstream 增加 24h 陈旧回退逻辑，读取并写入缓存第 6 行时间戳。

```
# check-upstream 作业中的陈旧回退与缓存写入逻辑（节选）
# 从缓存文件读取上次构建时记录的上游 SHA 与最后构建时间
# 第 6 行是本次新增的 last-build epoch 时间戳
LAST_SGLANG=$(sed -n '1p' "$CACHE_FILE")
LAST_MEGATRON=$(sed -n '2p' "$CACHE_FILE")
LAST_WHEELS_CU13_X86=$(sed -n '3p' "$CACHE_FILE")
LAST_WHEELS_CU13_ARM64=$(sed -n '4p' "$CACHE_FILE")
LAST_WHEELS_CU12_X86=$(sed -n '5p' "$CACHE_FILE")
LAST_BUILD_EPOCH=$(sed -n '6p' "$CACHE_FILE")

# … 上游 SHA 比较逻辑省略，其决定 SHOULD_BUILD 的初值 …

# miles 仓库自身的提交不会触发重建，所以用 24h 陈旧上限兜底：
# 只要距上次真正构建 ≥ 24h，即使上游无变化也强制重建，
# 保证 dev 镜像不会长期落后于 miles 仓库。
MAX_BUILD_AGE_SECONDS=$(( 24 * 3600 ))
NOW_EPOCH=$(date +%s)
if [ "$SHOULD_BUILD" = "false" ]; then
  # 旧版 5 行缓存没有时间戳，视为陈旧，重建一次并完成迁移
  if ! [ "$LAST_BUILD_EPOCH" -gt 0 ] 2>/dev/null; then
    SHOULD_BUILD=true
    echo "No last-build timestamp recorded; rebuilding"
  elif [ $(( NOW_EPOCH - LAST_BUILD_EPOCH )) -ge $MAX_BUILD_AGE_SECONDS ]; then
    SHOULD_BUILD=true
    echo "Last build was $(( (NOW_EPOCH - LAST_BUILD_EPOCH) / 3600 ))h ago (max $(( MAX_
    BUILD_AGE_SECONDS / 3600 ))h); rebuilding"
  fi
fi

# 缓存仅在构建触发时保存，所以第 6 行记录的是“最后一次真正构建”的时间
echo "${SGLANG_SHA}" > "$CACHE_FILE"
echo "${MEGATRON_SHA}" >> "$CACHE_FILE"
echo "${WHEELS_CU13_X86_FP}" >> "$CACHE_FILE"
echo "${WHEELS_CU13_ARM64_FP}" >> "$CACHE_FILE"
echo "${WHEELS_CU12_X86_FP}" >> "$CACHE_FILE"
echo "${NOW_EPOCH}" >> "$CACHE_FILE"
echo "should_build=${SHOULD_BUILD}" >> "$GITHUB_OUTPUT"
```

评论区精华

guapisolo 在 issue 评论中要求 "use doc-dev to modify docs"，强调文档变更应走 doc-dev 流程。Zhichenzzz 随即回复已更新 docs/ci/02-docker-build.md，说明 check-upstream 段落现在描述 24h 陈旧界限，触发器表行已改为 "build if upstream moved or last build ⩾ 24h ago"。PR body 还提到 24h 上限是在 build 频道讨论后由 guapisolo 确定的一天。最终 guapisolo 以 "LGTM" 批准合并，无其他 review 评论。

- 文档修改要求使用 doc-dev 流程 (documentation): 作者按要求完成文档更新，PR 获得批准。

风险与影响

- 风险：
  1. 构建频率上升：镜像最多每 24h 重建一次，比原先 " 仅上游变化 " 更频繁，会消耗更多构建资源与镜像仓库空间，但每天最多一次，成本可控。
  2. 缓存迁移兼容：旧 5 行缓存或损坏的第 6 行会被当作无时间戳，触发一次额外重建；[" $LAST_BUILD_EPOCH" -gt 0] 对非数字输入会静默失败并走迁移分支，行为符合预期，但增加了 " 多一次构建 " 的瞬时开销。
  3. 手动构建不刷新时间戳：workflow_dispatch / push-to-main 构建不写缓存，最坏情况下手动构建后下一次计划运行仍可能重建；这是有意设计，但需要团队知晓。
  4. 无自动化测试：该逻辑仅存在于 CI 脚本中，没有对应测试覆盖，未来修改需依赖 review 或手动验证。 - 影响：对用户：radixark/miles:dev 镜像将周期性刷新，避免长期缺失上游或 miles 自身更新；对系统：定时构建可能从 " 几乎不构建 " 变为 " 最多每天一次 "，增加构建队列与镜像仓库存储压力；对团队：需要知晓 24h 陈旧策略，并在修改构建逻辑时同步更新文档。 - 风险标记：构建频率增加，缓存格式迁移，无自动化测试

关联脉络

- PR #1961 docker: keep the cu12 dependency markers after checking out sglang-miles: 同为 docker 镜像构建链路的 CI 修复，改动 docker/Dockerfile 与镜像构建流程，与本 PR 的构建触发策略相关。
- PR #1795 Bump sglang to v0.5.16: 涉及 docker/Dockerfile 和 CI 工作流（.github/workflows/_run-ci.yml），整体影响镜像构建依赖栈。