

PR #53443 完整报告

vllm-project/vllm

[CI/Build][Hardware][NVIDIA] Add opt-in Rubin Docker builds

合并时间: 2026-08-28 00:06

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53443>

执行摘要

- 一句话: opt-in 支持 Rubin 预发布 Docker 构建, 默认路径不变
- 推荐动作: 推荐构建 / 部署团队精读 docker/Dockerfile 与两个 Triton 脚本, 重点学习几个设计: 如何用 `--mount=type=bind` 让私有清单不进入镜像层、如何用构建期环境变量绕过 pinned 工具链的旧汇编器而不污染运行时、以及上游仓库 `.gitignore` 对下游 fork 的连锁影响。这是一个『默认路径零影响 + opt-in 新路径』的较好范例。若你不在 NVIDIA 生态内, 该 PR 的运维价值有限, 但设计手法仍有参考意义。

功能与动机

NVIDIA Rubin 是新一代 GPU 架构 (VR200/R100), 需要 CUDA 13.4/13.5 与源码 Triton 才能编译运行, 但相关镜像和包仍是预发布输入, 无法进入公共 CI。PR body 明确说明该工作 Tracks #49735, 并强调与 #52379 (公开 CUDA-overlay/nightly 预览与预发布 CI) 相区分: 本 PR 不添加私有文件或 CI, 标准 Blackwell 镜像继续走公开依赖路径, 只对 NVIDIA 工程师和合作伙伴开放 opt-in 构建。

实现拆解

1. 新增 opt-in 构建参数与开关注册: docker/Dockerfile 顶部新增 `INSTALL_RUBIN_PRERELEASE=false`、`TRITON_INSTALL_FROM_SOURCE_REPO`、`TRITON_INSTALL_FROM_SOURCE_REVISION` 三个 ARG; docker/versions.json 同步注册 `INSTALL_RUBIN_PRERELEASE` 默认值 `false`, 保证版本配置与 Dockerfile 一致。
2. base 阶段 Rubin 依赖安装分支: 通过 `--mount=type=bind,source=requirements,target=/tmp/vllm-requirements,ro` 只读挂载调用方提供的 `requirements` 目录 (不拷入镜像层)。`INSTALL_RUBIN_PRERELEASE=true` 时先校验 `rubin-prerelease.txt` 存在, 缺失即退出; 随后按「Rubin 依赖 → 公开 cuda.txt 依赖 → Rubin 依赖」顺序安装, 私有清单不会进入公开镜像层。
3. 通用源码 Triton 构建 / 安装脚本: 新增 `tools/build_triton_from_source.sh` 与 `tools/install_triton_from_source.sh`, 在 `extensions-build` 阶段由 `TRITON_INSTALL_FROM_SOURCE_REPO` 非空触发。构建脚本按 `TARGETARCH` 选择 `almalinux-x64/almalinux-arm64` 的 LLVM 系统后缀, 浅克隆指定 `commit/branch/tag` (空 revision 默认 `main`), 用 `uv build --no-build-isolation` 产 wheel 并写 `wheel.sha256`; 安装脚本先校验 SHA-256、确认唯一 wheel, 再 `uv pip install --reinstall --no-deps`, 并按 `TRITON_SYSTEM_PTXAS_PATH` 链接 `ptxas-blackwell`, 用 `get_ptxas(107)` 断言版本

与路径。

4. 版本感知的工具链处理：CUDA 13.4 的预发布 PyTorch wheel 依赖 libcurand.so.11, Dockerfile 查找 libcurand.so.10 建立软链；CUDA 13.5 使用原生 libcurand.so.11。最终镜像把 CUDA 元数据规范到所选工具链版本，以通过 OCI 运行时检查；TRITON_PTXAS_BLACKWELL_PATH 仅在 wheel 构建 RUN 中设置，不持久化为运行时环境变量。
5. 收敛分支差异与文档配套：Rust 构建阶段无条件安装 make、perl-core，移除基于 Rubin 开关的分支；docs/getting_started/installation/gpu.cuda.inc.md 新增 Rubin 预览构建章节（CUDA 13.4/13.5 命令均使用 BUILD_BASE_IMAGE、FINAL_BASE_IMAGE 占位符），并澄清 torch_cuda_arch_list="" 委托 PyTorch 选架构的语义；dockerfile-stages-dependency.png 重新生成，dockerfile.md 同步新增参数说明。测试层面没有新增自动化用例，依赖 PR body 中列出的 Blackwell (GB300/B100) 与 Rubin (VR200/R100) 人工联调矩阵。

关键文件：

- docker/Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure；符号 INSTALL_RUBIN_PRERELEASE, TRITON_INSTALL_FROM_SOURCE_REPO, TRITON_INSTALL_FROM_SOURCE_REVISION）：核心变更：新增 INSTALL_RUBIN_PRERELEASE 等构建参数、base 阶段 Rubin 依赖校验与安装分支、版本感知的 cuRAND/PTXAS 处理、Rust 阶段统一安装 make/perl-core。
- tools/build_triton_from_source.sh（模块 构建脚本；类别 other；类型 core-logic）：新增通用源码 Triton 构建脚本，支持任意仓库 / 修订版本，处理 amd64/arm64 的 LLVM 系统后缀，是 Rubin 支持的关键配套。
- tools/install_triton_from_source.sh（模块 构建脚本；类别 other；类型 core-logic）：新增源码 Triton 安装脚本，校验 wheel 并处理 ptxas-blackwell 链接与版本断言，是 Rubin 工具链版本感知的核心实现。
- docs/getting_started/installation/gpu.cuda.inc.md（模块 安装文档；类别 docs；类型 documentation）：新增 Rubin 预览构建文档章节和占位符命令，并澄清 torch_cuda_arch_list 委托 PyTorch 的语义。
- docker/versions.json（模块 版本配置；类别 infra；类型 configuration）：注册 INSTALL_RUBIN_PRERELEASE 构建参数默认值，与 Dockerfile 保持一致。
- docs/assets/contributing/dockerfile-stages-dependency.png（模块 文档配图；类别 docs；类型 documentation）：根据合并后的 Dockerfile 重新生成 stage 依赖图。
- docs/contributing/dockerfile/dockerfile.md（模块 构建文档；类别 docs；类型 documentation）：同步记录新增构建参数，保持文档与 Dockerfile 一致。

关键符号：tools/build_triton_from_source.sh, tools/install_triton_from_source.sh

关键源码片段

tools/build_triton_from_source.sh

新增通用源码 Triton 构建脚本，支持任意仓库 / 修订版本，处理 amd64/arm64 的 LLVM 系统后缀，是 Rubin 支持的关键配套。

```

#!/usr/bin/env bash
set -euxo pipefail

# 必填参数: 目标架构与 Triton 仓库; revision 为空时默认用 main
: "${TARGETARCH:?TARGETARCH must be set}"
: "${TRITON_INSTALL_FROM_SOURCE_REPO:?TRITON_INSTALL_FROM_SOURCE_REPO must be set}"

TRITON_REVISION="${TRITON_INSTALL_FROM_SOURCE_REVISION:-main}"
TRITON_WORKSPACE="${TRITON_WORKSPACE:-/tmp/triton_from_source_workspace}"
TRITON_SOURCE="${TRITON_WORKSPACE}/src"
TRITON_DIST="${TRITON_WORKSPACE}/dist"
TRITON_BUILD_JOBS="${MAX_JOBS:-$(nproc)}"

# Triton 的 LLVM 依赖按系统架构选择预编译包
case "${TARGETARCH}" in
    amd64) TRITON_LLVM_SYSTEM_SUFFIX=almalinux-x64 ;;
    arm64) TRITON_LLVM_SYSTEM_SUFFIX=almalinux-arm64 ;;
    *) echo "Unsupported Triton build architecture: ${TARGETARCH}" >&2; exit 1 ;;
esac
export TRITON_LLVM_SYSTEM_SUFFIX

# 浅克隆指定修订 (commit/branch/tag), detach 后构建 wheel 并记录 SHA-256
mkdir -p "${TRITON_DIST}"
git init -q "${TRITON_SOURCE}"
git -C "${TRITON_SOURCE}" remote add origin "${TRITON_INSTALL_FROM_SOURCE_REPO}"
git -C "${TRITON_SOURCE}" fetch --depth=1 origin "${TRITON_REVISION}"
git -C "${TRITON_SOURCE}" checkout -q --detach FETCH_HEAD
git -C "${TRITON_SOURCE}" rev-parse HEAD

uv pip install --python /opt/venv/bin/python3 \
    -r "${TRITON_SOURCE}/python/requirements.txt"

TRITON_BUILD_WITH_CCACHE=true \
MAX_JOBS="${TRITON_BUILD_JOBS}" \
uv build --python /opt/venv/bin/python3 --wheel --no-build-isolation \
    --out-dir "${TRITON_DIST}" "${TRITON_SOURCE}"

(cd "${TRITON_DIST}" && sha256sum triton-*.whl > wheel.sha256)
rm -rf "${TRITON_SOURCE}"

```

tools/install_triton_from_source.sh

新增源码 Triton 安装脚本, 校验 wheel 并处理 ptxas-blackwell 链接与版本断言, 是 Rubin 工具链版本感知的核心实现。

```

#!/usr/bin/env bash
set -euxo pipefail

if [[ $# -ne 2 ]]; then

```

```

    echo "Usage: $0 <wheel-directory> <checksum-file>" >&2
    exit 1
fi

TRITON_DIST="$1"
TRITON_CHECKSUM="$2"

# 先校验 wheel 的 SHA-256, 再确认目录里恰好只有一个 Triton wheel
test -d "${TRITON_DIST}"
test -f "${TRITON_CHECKSUM}"
(cd "${TRITON_DIST}" && sha256sum -c "${TRITON_CHECKSUM}")

shopt -s nullglob
TRITON_WHEELS=("${TRITON_DIST}"/triton-*.whl)
if [[ ${#TRITON_WHEELS[@]} -ne 1 ]]; then
    echo "Expected exactly one Triton wheel in ${TRITON_DIST}" >&2
    exit 1
fi

uv pip install --system --reinstall --no-deps "${TRITON_WHEELS[0]}"

# Rubin 需要 SM107 汇编: pinned Triton 自带的 ptxas-blackwell 13.3.33 拒绝 sm_107a,
# 这里用 CUDA 13.4/13.5 系统 ptxas 覆盖包内链接, 并通过 get_ptxas(107) 断言版本与真实路径
if [[ -n "${TRITON_SYSTEM_PTXAS_PATH:-}" || -n "${TRITON_EXPECTED_PTXAS_VERSION:-}" ]]; then
    : "${TRITON_SYSTEM_PTXAS_PATH:?TRITON_SYSTEM_PTXAS_PATH must be set}"
    : "${TRITON_EXPECTED_PTXAS_VERSION:?TRITON_EXPECTED_PTXAS_VERSION must be set}"
    "

    PTXAS_PATH="${TRITON_SYSTEM_PTXAS_PATH}"
    test -x "${PTXAS_PATH}"
    TRITON_NVIDIA_BIN=$(python3 -c \
        'from pathlib import Path; import triton; print(Path(triton.__file__).parent / "backends" /
        "nvidia" / "bin")')
    mkdir -p "${TRITON_NVIDIA_BIN}"
    ln -sf "${PTXAS_PATH}" "${TRITON_NVIDIA_BIN}/ptxas-blackwell"

    env -u TRITON_PTXAS_BLACKWELL_PATH \
        TRITON_EXPECTED_PTXAS_PATH="${PTXAS_PATH}" \
        python3 -c \
        'import os; from triton.backends.nvidia.compiler import get_ptxas; tool = get_ptxas(107);
        assert tool.version == os.environ["TRITON_EXPECTED_PTXAS_VERSION"]; assert os.path.
        samefile(tool.path, os.environ["TRITON_EXPECTED_PTXAS_PATH"])'
fi

```

评论区精华

Review 主要由作者自己发现问题并修订: 其一, `TRITON_PTXAS_BLACKWELL_PATH` 是否仍需设置——结论是构建期仍需要, 因为 pinned Triton 的 `nvidia-toolchain-version.json` 绑定 `ptxas-blackwell 13.3.33` (拒绝 `sm_107a`), 且其构建 helper 只在提供该变量时才跳过捆绑

旧汇编器；该覆盖不持久化到运行时。其二，`requirements/rubin-prerelease.txt` 不能进 `.gitignore`，否则下游 `vllm-nr` 同步上游后私有清单会被意外忽略，已移除。其三，Rust 阶段接受无条件安装 `make perl-core`，以最小化 Rubin/Blackwell 构建差异。其四，安装脚本输入更名 `TRITON_SYSTEM_PTXAS_PATH`，防止与运行时 `override` 混淆。

- `TRITON_PTXAS_BLACKWELL_PATH` 构建期覆盖是否仍需要 (design): 保留该构建期覆盖并补文档；安装脚本输入重命名为 `TRITON_SYSTEM_PTXAS_PATH`，避免与运行时 `override` 混淆。
- `requirements/rubin-prerelease.txt` 不应加入 `.gitignore` (design): 已从 `.gitignore` 移除该条目，私有清单只保留在 `vllm-nr` 的 `staging` 分支。
- Rust 构建阶段是否应无条件安装 `make/perl-core` (design): 接受该方案，`rust-build` 不再声明或分支 `INSTALL_RUBIN_PRERELEASE`。
- 安装脚本变量命名避免与运行时 `override` 混淆 (design): 已改名并在注释与文档中澄清语义。

风险与影响

- 风险: Dockerfile 是全局构建路径: 新增 `bind mount` 校验与分支虽然默认关闭，但仍影响每次镜像构建，需要对默认 Blackwell 流程做回归 (PR body 显示 GB300/B100 已过)。源码 Triton 使用可变 `ref` (branch/tag) 时 `BuildKit` 不会自动失效缓存，必须 `--no-cache-filter extensions-build`；文档已提示，但自动化构建仍可能踩坑。wheel 大小方面，严格构建超过 500 MB 守卫，最终验证使用 `RUN_WHEEL_CHECK=false`，正式产物大小有待跟踪。驱动兼容性方面，CUDA 13.5 在 610.47.04 驱动上编译内核失败导致无法 `serving`，说明 Rubin 对驱动版本敏感。此外该路径无自动化测试，全部靠人工矩阵验证，且合并后的 `head` 只做了聚焦检查，未完整重建 CUDA 13.5 镜像；基础镜像的四组件 `CUDA_VERSION` 元数据属于上游缺陷，本 PR 只在输出侧做了归一化。
- 影响: 用户影响: 标准 Blackwell 镜像构建行为不变；NVIDIA 内部 / 合作伙伴可用 `opt-in` 参数构建 Rubin 镜像，并需要自行提供私有清单与基础镜像。系统影响: Docker 构建新增 3 个 ARG 与 2 个脚本，`stage` 图与文档同步更新；构建上下文必须包含 `requirements/` 目录 (`bind mount` 要求)。团队影响: 新增 Rubin 工具链 (PTXAS、cuRAND、元数据) 的维护分支，未来 released Triton 支持 SM107 后需移除 `package-local ptxas-blackwell` 链接这一临时集成。
- 风险标记: 构建路径变更需回归验证，依赖 NVIDIA 内部预发布包，源码 Triton 可变 `ref` 缓存失效，无自动化测试，靠人工矩阵验证，驱动版本兼容性差异

关联脉络

- PR #52379 (标题未提供，见 PR body 描述): PR body 明确区分: 该 PR 提供公开 `CUDA-overlay/nightly` 预览与预发布 CI，本 PR 只是 `opt-in` 私有构建路径，无私有文件或 CI。