PR #43292 完整报告

vllm-project/vllm

[CI] Pin protoc binary in rust-build stages

合并时间：2026-05-21 18:38

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43292

执行摘要

- 一句话：固定 Rust 前端构建的 protoc 版本，修复 manylinux 构建失败
- 推荐动作：值得关注 install_protoc.sh 的编写模式：通过独立脚本管理构建依赖，支持版本覆盖和环境检测。对于跨平台 Docker 构建，该模式可复用。安全校验的缺失属于既有权衡，可根据组织安全策略补充。

功能与动机

Rust 前端的 build.rs 传递了 --experimental_allow_proto3_optional 标志，要求 protoc >= 3.12。然而 manylinux2_28 构建器（RHEL/AlmaLinux 8）自带的 protoc 仅 3.5，导致构建失败。需要统一固定 protoc 版本以解决跨平台兼容性问题。

实现拆解

1. 创建 tools/install_protoc.sh 脚本，从 GitHub Releases 下载并安装固定版本（默认 34.2）的 protoc 二进制，支持架构映射和版本覆盖。
2. 修改 docker/Dockerfile、docker/Dockerfile.cpu、docker/Dockerfile.nightly_torch、docker/Dockerfile.rocm、docker/Dockerfile.xpu 共 5 个文件：移除 protobuf-compiler 和 libprotobuf-dev 系统包，安装 unzip 依赖，复制并运行 install_protoc.sh 脚本。
3. 脚本在容器中作为 root 运行，解压到 /usr/local。安装后清理临时文件。 无测试或配置配套变更。

关键文件：

- tools/install_protoc.sh（模块 工具脚本；类别 infra；类型 core-logic）：核心脚本，统一管理 protoc 安装逻辑
- docker/Dockerfile（模块 构建镜像；类别 infra；类型 infrastructure）：主要 Docker 构建文件，修改了 protoc 安装方式

关键符号：未识别

关键源码片段

tools/install_protoc.sh

核心脚本，统一管理 protoc 安装逻辑

```
#!/usr/bin/env bash
```

```
set -euo pipefail

# 从上游 GitHub releases 安装固定版本的 protoc 二进制。
#
# 发行版自带的 protobuf-compiler 版本差异很大（如 AlmaLinux/RHEL 8 提供的是
# protoc 3.5，早于 Rust 前端 build.rs 使用的 --experimental_allow_proto3_optional 标志），
# 因此我们在这里固定 protoc 版本。
# 可通过 PROTOC_VERSION 环境变量覆盖默认版本。
# 需要 curl、unzip 和 root 权限。

if [[ $(id -u) -ne 0 ]]; then
  echo "Must be run as root" >&2
  exit 1
fi

VERSION="${PROTOC_VERSION:-34.2}" # 默认版本 34.2，可覆盖

ARCH="$(uname -m)"
case "${ARCH}" in
  # protoc 发布包使用 aarch_64（下划线），不是 aarch64。请不要修改此 mapping。
  aarch64|arm64) URL_ARCH="aarch_64" ;;
  x86_64|amd64) URL_ARCH="x86_64" ;;
  *) echo "Unsupported arch for protoc binary: ${ARCH}" >&2; exit 1 ;;
esac

URL="https://github.com/protocolbuffers/protobuf/releases/download/v${VERSION}/protoc-
${VERSION}-linux-${URL_ARCH}.zip"
TMPDIR="$(mktemp -d)"
trap 'rm -rf "${TMPDIR}"' EXIT

echo "Downloading: ${URL}"
curl -fsSL -o "${TMPDIR}/protoc.zip" "${URL}" # 省略校验和，与其他安装脚本保持一致
unzip -q -o "${TMPDIR}/protoc.zip" -d /usr/local
echo "Installed $(protoc --version)"
```

docker/Dockerfile

主要 Docker 构建文件，修改了 protoc 安装方式

```
# 安装基本 C 工具链（一些 Rust crate 在 build.rs 中编译 C 代码）和 unzip（用于解压 protoc）
RUN if [ "${BUILD_OS}" = "manylinux" ]; then \
    dnf install -y --setopt=install_weak_deps=False \
        ca-certificates curl git gcc gcc-c++ make unzip \
    && dnf clean all && rm -rf /var/cache/dnf; \
  else \
    apt-get update -y \
    && apt-get install -y --no-install-recommends \
        ca-certificates curl git build-essential unzip \
    && rm -rf /var/lib/apt/lists/*; \
  fi
```

```
# 复制并运行固定版本 protoc 安装脚本（替代之前安装 protobuf-compiler 的步骤）
COPY tools/install_protoc.sh /tmp/install_protoc.sh
RUN /tmp/install_protoc.sh && rm /tmp/install_protoc.sh

# 安装 rustup（工具链由 rust/rust-toolchain.toml 固定）
RUN curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | \
    sh -s -- -y --profile minimal --default-toolchain none
```

评论区精华

- 版本号质疑：gemini-code-assist[bot] 指出默认版本 34.2 可能不存在（当时认为最新稳定版为 24.2）。但实际 v34.2 已发布，作者未修改。
- 安全审查：多位 reviewer 强调缺少 SHA256 校验和，存在供应链风险。作者回复“skip since other install scripts didn't do the check”，决定遵循现有模式不添加校验。
- 默认版本号 34.2 存疑 (correctness): 版本号保持不变，34.2 有效。
- 缺少校验和验证 (security): 作者回复 'skip since other install scripts didn't do the check'，决定不添加，与现有实践保持一致。

风险与影响

- 风险：
  1. 缺少校验和：下载的 protoc 二进制无完整性校验，若上游被篡改可能引入恶意代码。但现有 install_gdrcopy.sh 等脚本也未做校验，风险已在可接受范围。
  2. 版本固定风险：默认版本 34.2 未来可能被删除，但用户可通过 PROTOC_VERSION 环境变量覆盖。
  3. 网络依赖：构建时需访问 GitHub Releases，若网络不可达则构建失败（无显式重试）。 - 影响：影响所有使用 Rust 前端构建的 Docker 镜像，包括 CPU、GPU（CUDA）、ROCm、XPU 平台。修复了 release-v2 管道的构建失败，属于阻塞性修复。变更本身规模小，影响面广但风险低。 - 风险标记：缺少校验和，依赖外部下载，版本固定风险

关联脉络

- PR #40848 [Frontend][RFC] Rust front-end integration: 引入 Rust 前端构建，本 PR 修复其构建依赖问题