PR #52593 完整报告

vllm-project/vllm

[Build] Propagate vLLM version to Rust binaries

合并时间：2026-08-18 14:37

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52593

执行摘要

- 一句话：构建系统将 vLLM 源版本注入 Rust 二进制与版本接口
- 推荐动作：值得精读。该 PR 展示了跨语言构建链路中版本信息传递的完整设计：以 setuptools_scm 为单一事实来源，通过环境变量注入 Rust 编译期常量，并在 Docker 多阶段构建中优雅处理 Git 元数据缺失（.git 与 .git_archival.txt 兼容）与 partial worktree 误判 dirty 的问题。对于从事 vLLM 构建系统、Rust 前端或 Docker 镜像维护的工程师，本 PR 的 prepare_build_environment 函数和 build-info crate 模式可直接作为版本注入的参考实现。

功能与动机

PR body 明确指出：Rust artifacts currently identify themselves using workspace crate metadata, which stays at 0.1.0 and does not identify the corresponding nightly or release vLLM source revision. 这意味着用户无法从 Rust 侧工具版本追溯到对应的 vLLM 源码修订，影响排障和版本对齐。该 PR 在编译 Rust 前解析源版本，并通过 VLLM_RS_BUILD_VERSION 传入，同时让 get_vllm_version() 先于 Rust 构建执行，使 VLLM_VERSION_OVERRIDE 经 SETUPTOOLS_SCM_PRETEND_VERSION 同样作用于 Rust 工件。

实现拆解

1. 版本解析入口：在 tools/build_rust.py 中新增 prepare_build_environment()，优先读取 VLLM_RS_BUILD_VERSION 环境变量，否则调用 setuptools_scm.get_version(root=ROOT_DIR) 从 Git 元数据解析源版本，并将结果写回环境变量；main() 在 build_binary() 前调用该函数，让本地 bash build_rust.sh 和 setuptools-rust 构建路径都拿到版本。
2. Python 打包时序保证：在 setup.py 的 setup() 调用前先执行 vllm_version = get_vllm_version()，再执行 rust_build.prepare_build_environment()。这是因为 get_vllm_version() 可能设置 SETUPTOOLS_SCM_PRETEND_VERSION，而 prepare_build_environment() 内部依赖该变量来解析出与 Python 包一致的版本，保证 nightly/release wheel 构建中 Rust 版本与 Python 版本同步。
3. Rust 侧版本接入：新增 rust/src/build-info crate，在 lib.rs 中定义 pub const VERSION: &str，优先读取编译期环境变量 VLLM_RS_BUILD_VERSION，缺失时回退到 CARGO_PKG_VERSION（0.1.0）。随后将 vllm-build-info 加入 rust/Cargo.toml workspace，并为 rust/src/cmd/Cargo.toml、rust/src/bench/Cargo.toml 添加依赖；

vllm-rs 和 vllm-bench 的 clap #[command(version = vllm_build_info::VERSION)] 改为使用该常量，rust/src/server/src/routes/version.rs 的 rust_frontend_version 字段同样改为该常量，引擎上报的 version 字段保持不变。

4. Docker 构建路径适配：在 docker/Dockerfile、.cpu、.rocm、.rocm_gfx1250、.xpu 中增加获取源码版本逻辑。由于 selective-copy 阶段会省略部分 tracked 文件，若不处理会让 Git 错误地把 partial worktree 标记为 dirty，因此构建阶段显式设置 dirty=false。ROCm 场景因镜像可能不携带 .git（为减小体积和优化缓存），BugenZhao 新增了一个轻量共享stage，调用setuptools_scm自动区分git存在或仅存在git_archival.txt的情况。.buildkite/scripts/ci-bake-rocm.sh 同步调整了 bake 时的版本解析参数。
5. 测试与验证配套：tests/entrypoints/openai/test_uds.py 和 tests/entrypoints/serve/instrumentator/test_basic.py 在启用 VLLM_USE_RUST_FRONTEND 时新增断言 rust_frontend_version == VLLM_VERSION；Rust 端 rust/src/server/src/routes/tests.rs 的版本路由测试改为断言 vllm_build_info::VERSION。作者在 CI 中验证 CUDA/ROCm/CPU 镜像的 Rust 二进制均报告 0.27.2rc1.dev192+g2db2a2484，准确对应源提交。

关键文件：

- tools/build_rust.py（模块 构建工具；类别 source；类型 dependency-wiring；符号 prepare_build_environment）：新增 prepare_build_environment()，是版本解析与注入的入口，负责从 setuptools-scm 解析源版本并写入 VLLM_RS_BUILD_VERSION 环境变量，决定整个构建链的版本来源。
- rust/src/build-info/src/lib.rs（模块 构建信息；类别 source；类型 core-logic）：新增的 vllm-build-info crate 是 Rust 侧版本暴露的核心，通过 VERSION 常量统一了 vllm-rs、vllm-bench 和 Rust 前端 /version 接口的版本来源。
- setup.py（模块 打包脚本；类别 source；类型 core-logic）：调整 Python 打包时序：先调用 get_vllm_version() 再调用 prepare_build_environment()，确保 VLLM_VERSION_OVERRIDE 经 SETUPTOOLS_SCM_PRETEND_VERSION 传递给 Rust 构建，是 wheel 构建路径版本一致性的关键。
- rust/src/cmd/src/cli.rs（模块 命令行入口；类别 source；类型 core-logic）：vllm-rs 的 clap 命令版本由默认 crate 版本改为 vllm_build_info::VERSION，使 vllm-rs --version 输出真实 vLLM 源版本。
- rust/src/bench/src/main.rs（模块 基准工具；类别 source；类型 core-logic）：vllm-bench 的 clap 版本由 version（自动使用 crate 版本）改为显式绑定 vllm_build_info::VERSION，使基准工具报告源版本。
- rust/src/server/src/routes/version.rs（模块 版本接口；类别 source；类型 entrypoint）：Rust 前端 GET /version 的 rust_frontend_version 字段从 CARGO_PKG_VERSION 改为 vllm_build_info::VERSION，是用户可观测的版本接口。
- docker/Dockerfile.rocm（模块 镜像构建；类别 infra；类型 infrastructure）：ROCm 镜像可能不携带 .git，本 PR 在这里新增轻量 setuptools-scm 共享 stage 以兼容 .git / .git_archival.txt 两种元数据来源，并设置 dirty=false，是讨论中重点关注的改动。
- tests/entrypoints/openai/test_uds.py（模块 端到端测试；类别 test；类型 test-coverage）：E2E 测试在 Rust 前端启用时新增 rust_frontend_version == VLLM_VERSION 断言，验

证版本注入端到端生效。

关键符号：prepare_build_environment

关键源码片段

tools/build_rust.py

新增 prepare_build_environment()，是版本解析与注入的入口，负责从 setuptools-scm 解析源版本并写入 VLLM_RS_BUILD_VERSION 环境变量，决定整个构建链的版本来源。

```
# tools/build_rust.py 中新增的版本解析与注入逻辑

VLLM_RS_BUILD_VERSION = "VLLM_RS_BUILD_VERSION"


def prepare_build_environment() -> str | None:
    """设置 Rust 工件使用的设备无关 vLLM 源版本。"""
    # 优先读取外部显式传入的版本，例如 CI 通过环境变量指定
    version = os.getenv(VLLM_RS_BUILD_VERSION) or None
    if version is None:
        try:
            # 从 Git 元数据或 .git_archival.txt 解析源版本
            version = get_version(root=ROOT_DIR)
        except LookupError:
            # 既无 .git 也无 .git_archival.txt 时返回 None，
            # 调用方将回退到 Rust crate 版本（0.1.0）
            return None

    # 写回环境变量，供 setuptools-rust / cargo 在编译期读取
    os.environ[VLLM_RS_BUILD_VERSION] = version
    return version


def main() -> None:
    # 必须在 build_binary 之前解析版本，
    # 否则 RustExtension 编译时拿不到 VLLM_RS_BUILD_VERSION
    prepare_build_environment()
    build_binary(sys.argv[1:])
```

rust/src/build-info/src/lib.rs

新增的 vllm-build-info crate 是 Rust 侧版本暴露的核心，通过 VERSION 常量统一了 vllm-rs、vllm-bench 和 Rust 前端 /version 接口的版本来源。

```
// rust/src/build-info/src/lib.rs

/// 构建系统注入的 vLLM 源版本。
///
/// 直接使用 `cargo build` 且未设置 `VLLM_RS_BUILD_VERSION` 时，
/// 回退到 workspace crate 版本（0.1.0）。
```

```
pub const VERSION: &str = match option_env!("VLLM_RS_BUILD_VERSION") {
    Some(version) => version,
    None => env!("CARGO_PKG_VERSION"),
};
```

评论区精华

核心讨论围绕 ROCm 镜像的 Git 元数据可用性展开。AndreasKaratzas 提出担忧："The problem I think is that ROCm may not ship with .git such that we reduce size and also optimize docker caching"，即 ROCm 镜像为减小体积、优化缓存通常不携带 .git，可能导致版本解析失效。BugenZhao 回应已增加轻量共享 stage："I've managed to add a lightweight shared stage for retrieving the version by calling setuptools_scm there in ROCm Dockerfile, which should automatically handle different cases like when .git is present or from .git_archival.txt"，并补充验证结果 "Verified that the Rust binaries from all images here (CUDA, ROCm, CPU) correctly reports 0.27.2rc1.dev192+g2db2a2484"。AndreasKaratzas 最终确认 "Yep it is indeed lightweight. ROCm modification LGTM"。该讨论已解决，无未关闭疑虑。

- ROCm 镜像可能不携带 .git 导致版本解析失败 (design): BugenZhao 在 ROCm Dockerfile 中新增轻量共享 stage，调用 setuptools_scm 自动处理 .git 存在或仅存在 .git_archival.txt 的情况；验证 CUDA/ROCm/CPU 三类镜像均正确报告 0.27.2rc1.dev192+g2db2a2484 后，AndreasKaratzas 确认 ROCm 改动 LGTM。
- 跨平台版本一致性验证 (testing): 验证通过，三个平台输出一致，确认设备无关设计成立。
- pre-commit 检查与 CI 触发 (other): 经过多轮修复和重新触发，CI 全部通过。

风险与影响

- 风险：主要风险集中在构建链路和环境依赖上：1）若构建环境既无 .git 也无 .git_archival.txt，setuptools_scm 会抛 LookupError，prepare_build_environment() 返回 None，Rust 版本将回退到 crate 版本 0.1.0，与 Python 包版本不一致，可能误导用户；2）Docker selective-copy 阶段设置 dirty=false 会压制 partial worktree 的 dirty 标记，使版本号不含 dirty 后缀，精度略有损失；3）VLLM_RS_BUILD_VERSION 是新的构建期环境变量，直接 cargo build 时若未显式设置且没有 Git 元数据，输出仍为 0.1.0，与 wheel 行为存在差异；4）本次改动覆盖 5 个 Dockerfile（Dockerfile、.cpu、.rocm、.rocm_gfx1250、.xpu）及 bake 脚本，任何平台镜像构建路径的调整都可能导致构建失败或版本错误，依赖 CI 完整验证；5）重新打包预编译 Rust 二进制会保留构建时嵌入的版本，若二进制与 wheel 来自不同构建，版本可能不完全同步，属于已知且可接受的行为。运行时性能无影响。
- 影响：对用户：vllm-rs --version、vllm-bench --version 和 Rust 前端 GET /version 的 rust_frontend_version 字段将报告真实 vLLM 源版本，便于在问题反馈和日志排查中快速定位对应的源码修订。对系统：无运行时行为变化，仅影响构建期和版本报告路径；Python 分发版本（含 CPU/CUDA/ROCm/TPU/XPU 平台后缀）保持不变。对团队：统一了 Python 与 Rust 工件的版本语义，消除了 Rust 侧长期停留在 0.1.0 的盲区；同时为 Docker 多平台构建引入了可复用的版本解析 stage，后续新增平台镜像时可复用该模式。影响范围覆盖所有发布构建产物，属于基础设施层的系统性改进，但风险可控。

- 风险标记：多平台 Dockerfile 变更，Git 元数据缺失时版本回退歧义，partial worktree dirty 状态被压制，构建期环境变量依赖，预编译二进制版本与 wheel 可能不一致

关联脉络

- PR #52575 [Rust Frontend] Simplify data-parallel size ownership: 同为 Rust 前端构建与协议调整，修改了 rust/src/server 相关代码，与本 PR 在 Rust 前端构建链路上有交集，且都涉及发布二进制与版本 / 配置的传递。
- PR #52031 [Rust Frontend][gRPC] Advertise LoRA capabilities: 通过握手协议在 Rust 前端与 Python 引擎之间传递能力信息，与本 PR 通过构建环境变量传递版本信息属于同一类跨语言构建 / 协议元数据同步问题。
- PR #51208 [ROCm][AMD][Installation] add LMCache kv-connector installation and runtime packages to docker image: 修改了 docker/Dockerfile.rocm 和 .buildkite/scripts/ci-bake-rocm.sh，与本 PR 在 ROCm Docker 构建路径上直接重叠，属于同一镜像构建链路的演进。