PR #46696 完整报告

vllm-project/vllm

[Rust Frontend] Switch `rustls` to `native-tls`/OpenSSL

合并时间：2026-06-26 08:19

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46696

执行摘要

- 一句话：将 Rust 前端的 TLS 实现从 rustls 切换到 native-tls/OpenSSL，并添加依赖禁止检查。
- 推荐动作：该 PR 值得关注其依赖治理策略，特别是通过 cargo-deny 建立禁止列表和利用 Cargo feature unification 确保 TLS 一致性的方法。适合作为基础设施团队依赖管理的参考案例。

功能与动机

PR body 明确指出：'Eliminate rustls from the Rust frontend dependency tree and keep the HTTP/TLS stack on native-tls / OpenSSL, which is friendlier for compliance-sensitive deployments.' 此外还引用了 issue #46052 和 PR #45890 的讨论。

实现拆解

1. 调整工作空间依赖配置：在 rust/Cargo.toml 中，将 reqwest、async-openai、openai-harmony 的 TLS 后端从 rustls-tls 切换为 native-tls；禁用 hf-hub 的默认特性并仅启用 tokio 特性，避免其传递引入 rustls；升级 fastokens 到 0.2.1 并禁用默认特性，消除其对同步 hf-hub API 的依赖；更新 llm-multimodal 到新 commit 以获取 native-tls 支持。
2. 适配 tokenizer 基准测试：rust/src/tokenizer/benches/tiktoken.rs 和 rust/src/tokenizer/benches/hf.rs 中的模型加载函数 tiktoken_model 和 tokenizer_json 原先使用 hf_hub::api::sync::ApiBuilder 同步 API，现改为使用 hf_hub::api::tokio::ApiBuilder 并通过 tokio::runtime::Runtime::block_on 在同步上下文中执行异步调用。
3. 新增依赖禁止配置：创建 rust/deny.toml，配置 cargo-deny 的 bans 检查，禁止引入 rustls、ring、aws-lc-rs、aws-lc-sys、s2n-tls、s2n-tls-sys、boring、boring-sys 等 crate。
4. 集成 CI 检查：在 .buildkite/scripts/run-rust-frontend-cargo-ci.sh 中新增 install_cargo_deny 函数，并在 run_style_clippy 步骤中添加 cargo deny check bans 命令，确保每次 CI 都会执行依赖禁止检查。
5. 更新 lock 文件：rust/Cargo.lock 删除大量与 rustls 相关的包条目，反映依赖树的净减少（+21/-417）。

关键文件：

- rust/src/tokenizer/benches/tiktoken.rs（模块 tokenizer 测试；类别 source；类型 dependency-wiring；符号 tiktoken_model）：此基准测试因 hf-hub API 从同步改为异步而调整，使用 tokio runtime 包裹调用，体现了依赖变更对测试代码的影响。
- rust/src/tokenizer/benches/hf.rs（模块 tokenizer 测试；类别 source；类型 dependency-wiring；符号 tokenizer_json）：与文件 1 类似，将同步 hf-hub API 改为异步，影响 HuggingFace tokenizer 基准测试加载模型文件的方式。
- rust/deny.toml（模块 Rust 配置；类别 config；类型 configuration）：新文件，配置 cargo-deny 的 bans 规则，禁止引入与 native-tls 冲突的 TLS crate，是本次变更的治理核心。
- rust/Cargo.toml（模块 Rust 配置；类别 config；类型 configuration）：工作空间依赖配置的主文件，核心变更都在这里：切换 reqwest 等 crate 的 TLS 特性，修改 fastokens、hf-hub、openai-harmony、async-openai 等的依赖声明，是本次 PR 的核心配置变更。
- .buildkite/scripts/run-rust-frontend-cargo-ci.sh（模块 CI 脚本；类别 other；类型 core-logic；符号 install_cargo_deny, run_style_clippy）：CI 脚本，新增 cargo-deny 安装和运行步骤，将依赖禁止检查纳入 CI 流程，防止未来回归。

关键符号：install_cargo_deny, tiktoken_model, tokenizer_json

关键源码片段

rust/deny.toml

新文件，配置 cargo-deny 的 bans 规则，禁止引入与 native-tls 冲突的 TLS crate，是本次变更的治理核心。

```
# 允许同一个 crate 的多个版本（仅用于过渡，未来应设为 deny）
multiple-versions = 'allow'

# 禁止引入以下 TLS/crypto 库，以保持唯一 native-tls 后端
deny = [
    { name = 'rustls' }, # 纯粹的 Rust TLS 实现
    { name = 'ring' }, # rustls 底层加密库
    { name = 'aws-lc-rs' }, # AWS-LC 的 Rust 绑定
    { name = 'aws-lc-sys' }, # AWS-LC 系统库
    { name = 's2n-tls' }, # AWS s2n-tls 实现
    { name = 's2n-tls-sys' }, # s2n-tls 系统库
    { name = 'boring' }, # BoringSSL 的 Rust 绑定
    { name = 'boring-sys' }, # BoringSSL 系统库
]
```

评论区精华

> 供应链风险：depthfirst-app[bot] 指出 openai-harmony 从 crates.io 切换为第三方 git fork 可能破坏完整性验证。 TLS 完整性：chatgpt-codex-connector[bot] 指出 hf-hub 禁用默认特性后可能缺失 TLS。作者 BugenZhao 回复通过确保 vllm-text 和 vllm-tokenizer 显式依赖 reqwest/native-tls 利用特性统一解决。 运行时依赖：Harry-Chen 担忧 libssl.so 动态链接。作者确认 native-tls-vendored 静态链接无运行时依赖。

- 引入 openai-harmony git 依赖的供应链风险 (security): PR 最终合并，说明团队评估后接受了此风险，或已确认 fork 可信。
- hf-hub default-features=false 可能导致 TLS 缺失 (correctness): 作者 BugenZhao 回复同意，并指出不能直接在 hf-hub 上添加 native-tls 特性（因为会引入 ureq 和 rustls），而是需要在 vllm-text 和 vllm-tokenizer 中显式依赖 reqwest/native-tls，利用 Cargo 特性统一机制。后续 commit 中已确保 hf-hub 独立 TLS 后端。
- 对 libssl.so 运行时依赖的担忧 (question): 作者回复已启用 native-tls-vendored 特性，静态链接 OpenSSL，因此不会产生动态库依赖。

风险与影响

- 风险：
  - 供应链风险：openai-harmony 改用第三方 git fork（Inferact/openai-harmony），pinned commit 可能不可追溯，增加安全审计负担。
  - 构建体积：native-tls-vendored 静态链接 OpenSSL 会增大 Rust 二进制体积，但影响有限。
  - 依赖治理维护：deny.toml 禁止列表需要随依赖更新维护，可能误拦合法依赖（如未来若有合理使用 rustls 的场景）。
  - 基准测试变化：tokenizer bench 从同步改为异步调用，可能引入微小延迟，但 block_on 仅在初始化时执行一次，对基准影响可忽略。
  - 影响：影响范围局限于 Rust 前端构建产物（vllm-chat、vllm-text、vllm-tokenizer 等 crate），Python 后端无影响。对用户完全透明，但使 Docker 镜像和 wheel 中的 TLS 实现统一为 OpenSSL，有利于合规审核。对开发团队，需要维护 deny.toml 并在添加新依赖时确认 TLS 后端类型。
  - 风险标记：供应链风险（git 依赖），构建体积增大（vendored OpenSSL），依赖治理维护成本

关联脉络

- PR #46052：PR body 引用此 issue 作为相关讨论，涉及 TLS 选择的决策背景。
- PR #45890：PR body 引用此 PR 的 comment（#issuecomment-4783509287）作为相关讨论。