PR #43854 完整报告

vllm-project/vllm

[Rust Frontend] Add `/version` endpoint using engine-reported value

合并时间：2026-05-29 08:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43854

执行摘要

本 PR 在 Rust 前端新增 /version 端点，引擎通过握手协议上报 vLLM 版本，Rust 前端直接读取并返回给调用方。同时简化了 ready_response 的 Option 类型，移除了与旧引擎的兼容代码，使客户端 API 更加简洁。

功能与动机

Rust 前端需要知晓引擎的 vLLM 版本以进行行为适配（例如切换特性）。原有方案尝试直接从 Rust 导入 Python 全局变量，但不可靠且与引擎生命周期解耦。作者通过扩展握手消息中的 EngineCoreReadyResponse，让引擎在就绪时主动报告版本，Rust 前端作为第一类信息使用，彻底解决了版本获取的脆弱性。

实现拆解

1. Python 引擎侧新增字段：在 vllm/v1/engine/__init__.py 的 EngineCoreReadyResponse 中增加 vllm_version 字段，利用 vllm.__version__ 填充。
2. Rust 协议定义更新：在 rust/src/engine-core-client/src/protocol/handshake.rs 的 EngineCoreReadyResponse 结构体中新增 vllm_version: String，同时将 dtype 从 Option 提升为必选。
3. Rust 传输层重构：在 rust/src/engine-core-client/src/transport.rs 中，将 ConnectedEngine.ready_response 从 Option 改为直接持有，并重构 wait_for_input_registrations 使其返回 Vec<ConnectedEngine>，移除对旧引擎空注册消息的兼容支持。
4. Rust 客户端暴露访问器：在 rust/src/engine-core-client/src/client.rs 中添加 vllm_version() 方法，并从第一个引擎的响应中获取版本；由于 ready_response 不再可选，model_dtype()、max_model_len() 等方法也相应简化，不再返回 Option。
5. 新增 /version 路由与集成测试：在 rust/src/server/src/routes/version.rs 中实现 GET /version 处理函数，返回包含 version 和 rust_frontend_version 的 JSON；在 tests.rs 中添加集成测试，用 mock 引擎验证返回内容。

rust/src/engine-core-client/src/protocol/handshake.rs

协议数据结构的核心变更：新增 vllm_version 字段并将 dtype 从 Option 提升为必选，是版本功能实现的基础。

```
/// Post-initialization configuration sent from each engine on the input socket
/// registration message, after the handshake completes.
```

```
///
/// Contains values that may differ from the original config (e.g.
/// `max_model_len` after KV cache auto-fitting, `num_gpu_blocks` after
/// profiling).
///
/// Original Python definition:
/// <https://github.com/vllm-project/vllm/blob/c8d98f81f6/vllm/v1/engine/__init__.py#L67-L77>
#[derive(Debug, Clone, Serialize, Deserialize)]
pub struct EngineCoreReadyResponse {
    /// Engine-reported maximum model context length (auto-fitted after
    /// KV cache profiling and may differ from the original config value).
    pub max_model_len: u64,
    /// Number of GPU blocks available for KV cache on this engine.
    pub num_gpu_blocks: u64,
    /// DP coordinator stats publish address, if applicable.
    pub dp_stats_address: Option<String>,
    /// Effective model dtype after Python vLLM resolves `--dtype`.
    // 不再可选，引擎必须提供 dtype
    pub dtype: ModelDtype,
    /// Python vLLM version reported by the engine process.
    // 新增字段，引擎返回的 vLLM 版本字符串
    pub vllm_version: String,
}
```

rust/src/engine-core-client/src/client.rs

客户端 API 的简化与版本访问方法添加：vllm_version() 成为公开方法，同时多个查询方法移除 Option 返回类型。

```
    /// Return the ready responses received from all engines on the input
    /// socket.
    pub fn ready_responses(&self) -> Vec<&EngineCoreReadyResponse> {
        self.engines.iter().map(|engine| &engine.ready_response).collect()
    }

    /// Return the engine-reported effective model dtype.
    // 从第一个引擎的响应中直接获取 dtype，不再返回 Option
    pub fn model_dtype(&self) -> ModelDtype {
        self.engines
            .first()
            .expect("engine core client requires at least one engine")
            .ready_response
            .dtype
    }

    /// Return the engine-reported Python vLLM version.
    pub fn vllm_version(&self) -> &str {
        self.engines
            .first()
            .expect("engine core client requires at least one engine")
```

```
            .ready_response
            .vllm_version
            .as_str()
    }
```

rust/src/server/src/routes/version.rs

新增的 /version 路由处理函数，是该 PR 面向用户的核心产出。

```rust
use std::sync::Arc;

use axum::Json;
use axum::extract::State;
use serde::Serialize;

use crate::state::AppState;

/// 版本响应结构体，包含引擎版本和 Rust 前端版本
#[derive(Serialize)]
pub(crate) struct VersionResponse {
    /// 引擎 vLLM 版本（从 Python 引擎获取）
    version: String,
    /// Rust 前端 Cargo 包版本（编译时注入）
    rust_frontend_version: &'static str,
}

/// 处理 GET /version 请求，返回版本元信息
pub async fn version(State(state): State<Arc<AppState>>) -> Json<VersionResponse> {
    // 从引擎客户端获取 vLLM 版本
    let version = state.engine_core_client().vllm_version().to_string();

    Json(VersionResponse {
        version,
        // `CARGO_PKG_VERSION` 在编译时由 Cargo 注入
        rust_frontend_version: env!("CARGO_PKG_VERSION"),
    })
}
```

评论区精华

无审查讨论，PR 由维护者 @njhill 直接批准。作者在提交中逐步简化 Option 兼容代码后添加版本字段。

风险与影响

- 兼容性风险：ready_response 类型收紧为必选，旧版本引擎未发送注册消息将导致 panic，但 vLLM 的当前引擎版本总是发送。同步部署 Python 和 Rust 即可。
- 契约变更：dtype 必选化，要求 Python 引擎始终设置该字段。PR 中已确保。
- 测试覆盖：新增集成测试验证端点返回正确版本，但缺乏引擎未发送版本字段的异常测试。
- 性能影响：极小，减少一层 Option 过滤。

关联脉络

本 PR 是 Rust 前端功能增强的一部分，无直接关联 Issue。它揭示了 Rust 前端与 Python 引擎通过握手协议交换元信息的模式，类似的做法可推广到其他配置和状态查询。