PR #47741 完整报告

vllm-project/vllm

[Rust Frontend] Add Seed-OSS tool parser

合并时间：2026-07-16 17:28

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47741

执行摘要

本 PR 在 Rust 前端中为 Seed-OSS 模型添加了 tool-call 解析支持。通过将 Qwen3CoderToolParser 的包装标记参数化，以极少的代码复用了现有解析逻辑，实现了 Seed-OSS 专有 <seed:tool_call> 标记的解析。同时注册了模型名称模式，使该解析器可被自动选取。变更简洁、风险低，是典型的配置驱动复用模式。

功能与动机

Seed-OSS 模型的 tool-call 格式与 Qwen3-Coder 仅在包装标记上不同（<seed:tool_call> vs <tool_call>），内部<function=>/<parameter=>语法和schema驱动的参数转换完全一致。为避免重复实现，PR 设计为参数化现有解析器，而不是创建新的独立解析器。PR body 明确指出："I didn't add a new parser. I made the wrapper markers configurable on Qwen3CoderToolParser and added a thin SeedOssToolParser that points at it with the seed markers." 同时与 Python 前端实现 SeedOssParser(Qwen3Parser) 保持一致。

实现拆解

1. 参数化 Qwen3CoderToolParser（rust/src/parser/src/tool/qwen_coder.rs）：
   - 新增 QwenCoderConfig 结构体，包含 parser_name、tool_call_start、tool_call_end 字段。
   - 将原先硬编码的 TOOL_CALL_START / TOOL_CALL_END 作为 Qwen 默认配置。
   - 新增 with_config 构造函数，允许调用方指定自定义包装标记。
   - 修改 parse_next_qwen_coder_event 和 finish 中的错误检查，使用配置中的标记而非常量。
2. 实现 SeedOssToolParser（rust/src/parser/src/tool/seed_oss.rs，新增 253 行）：
   - 定义 SeedOssToolParser 结构体，内部持有 Qwen3CoderToolParser 实例。
   - 通过 with_config 传入 SEED_OSS_CONFIG（parser_name: "Seed-OSS", tool_call_start: "<seed:tool_call>", tool_call_end: "</seed:tool_call>"）。
   - 所有 ToolParser trait 方法（create、parse_into、finish、reset）直接委托给内部解析器。
   - 说明 structural_tag_model 返回 None，与 Python 的 SeedOssEngineToolParser 一致，不做 guided decoding。
   - 包含 4 个单元测试（纯文本留过、普通工具标记不触发、完整工具提取、流式测试）。

3. 注册解析器（rust/src/chat/src/parser/tool/mod.rs）：
   - 导入 SeedOssToolParser。
   - 添加 names::SEED_OSS 常量。
   - 注册解析器和模型匹配模式 "seed-oss" 和 "seedoss"，使 Auto 路由生效。
4. 更新测试与快照：
   - rust/src/chat/src/parser/tool/tests.rs：验证模型名称 "ByteDance-Seed/Seed-OSS-36B-Instruct" 可路由到 seed_oss。
   - rust/src/chat/src/lib.rs：更新 expect_test 中错误消息快照，在可用解析器列表中增加 seed_oss。
   - rust/src/chat/tests/roundtrip.rs：同步更新 roundtrip 测试快照。

rust/src/parser/src/tool/seed_oss.rs

新增核心文件，定义 SeedOssToolParser 结构体，通过配置复用 Qwen3CoderToolParser，并包含单元测试。

```
// SPDX-License-Identifier: Apache-2.0
// SPDX-FileCopyrightText: Copyright contributors to the vLLM project

use super::qwen_coder::{Qwen3CoderToolParser, QwenCoderConfig};
use crate::tool::{Result, Tool, ToolParser, ToolParserOutput};

// 定义 Seed-OSS 专用的配置，使用 <seed:tool_call> 作为包装标记
const SEED_OSS_CONFIG: QwenCoderConfig = QwenCoderConfig {
    parser_name: "Seed-OSS",
    tool_call_start: "<seed:tool_call>",
    tool_call_end: "</seed:tool_call>",
};

/// Seed-OSS 的 tool parser，内部分配给 Qwen3CoderToolParser
/// 内部 <function=...> / <parameter=...> 与 Qwen3-Coder 完全相同
pub struct SeedOssToolParser {
    inner: Qwen3CoderToolParser,
}

impl SeedOssToolParser {
    // 使用 SEED_OSS_CONFIG 创建内部解析器
    fn new(tools: &[Tool]) -> Self {
        Self {
            inner: Qwen3CoderToolParser::with_config(tools, SEED_OSS_CONFIG),
        }
    }
}

impl ToolParser for SeedOssToolParser {
    fn create(tools: &[Tool]) -> Result<Box<dyn ToolParser>> {
        Ok(Box::new(Self::new(tools)))
```

```
    }

    // 所有解析方法直接委托给 inner
    fn parse_into(&mut self, chunk: &str, output: &mut ToolParserOutput) -> Result<()> {
        self.inner.parse_into(chunk, output)
    }

    fn finish(&mut self) -> Result<ToolParserOutput> {
        self.inner.finish()
    }

    fn reset(&mut self) -> String {
        self.inner.reset()
    }
}
```

rust/src/parser/src/tool/qwen_coder.rs

核心修改文件，引入 QwenCoderConfig 配置结构体，使工具调用标记可参数化，影响所有使用 Qwen 格式的解析器。

```
/// 模型专用配置，用于共享的 Qwen Coder 语法
/// 不同模型仅包装标记不同，内部 <function=> / <parameter=> 标签总是相同
#[derive(Debug, Clone, Copy)]
pub(crate) struct QwenCoderConfig {
    pub(crate) parser_name: &'static str,
    pub(crate) tool_call_start: &'static str,
    pub(crate) tool_call_end: &'static str,
}

// Qwen 自身的默认配置
const QWEN_CODER_CONFIG: QwenCoderConfig = QwenCoderConfig {
    parser_name: "Qwen Coder",
    tool_call_start: TOOL_CALL_START, // "<tool_call>"
    tool_call_end: TOOL_CALL_END, // "</tool_call>"
};

impl Qwen3CoderToolParser {
    // 对外公开的构造方法，接收任意配置
    pub(crate) fn with_config(tools: &[Tool], config: QwenCoderConfig) -> Self {
        Self {
            buffer: String::new(),
            mode: QwenCoderMode::Text,
            emitted_tool_count: 0,
            tool_parameters: ToolSchemas::from_tools(tools),
            config,
        }
    }
}
```

rust/src/chat/src/parser/tool/mod.rs

注册 SeedOssToolParser 及其模型名称模式，使自动路由生效。

```
// 在 pub use 中添加 SeedOssToolParser
pub use vllm_parser::tool::{
    // ...
    Qwen3CoderToolParser,
    Qwen3XmlToolParser,
    SeedOssToolParser,
    ToolParser,
    ToolParserError,
};

// 在 names 模块中添加常量
pub const SEED_OSS: &str = "seed_oss";

// 在工厂注册中添加解析器和模式
factory
    .register_parser::<SeedOssToolParser>(names::SEED_OSS)
    .register_pattern("seed-oss", names::SEED_OSS)
    .register_pattern("seedoss", names::SEED_OSS);
```

评论区精华

审阅中无实质性讨论。BugenZhao 直接批准。作者在评论中说明“Adds seed_oss to the tool-parser name assertion CI flagged”，第二笔提交修复了 CI 断言，无功能变更。

风险与影响

风险：该项变更风险低。参数化配置若使用不当（如传递无效标记）可能导致解析失败；structural_tag_model 为 None 意味着 guided decoding 不可用，但 Python 端同样如此；模型模式“seed-oss”和“seedoss”匹配冲突概率很小。

影响：Seed-OSS 模型用户现在可以在 Rust 前端使用 tool-call 功能；内部架构无副作用；该配置复用模式可推广到其他只有包装标记不同的模型。

关联脉络

本 PR 是 Seed-OSS 支持 roadmap（#44280）的 tool-parser 部分。之前已有 Seed-OSS 推理解析器（SeedOssReasoningParser）和 Python 端的 SeedOssEngineToolParser。本 PR 完成后，Rust 前端对 Seed-OSS 的 tool-call 解析能力与 Python 前端对齐。