PR #43582 完整报告

vllm-project/vllm

[Rust Frontend] Add reasoning/tool parser & renderer roundtrip tests

合并时间：2026-05-27 08:49

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43582

执行摘要

该 PR 为 Rust 聊天前端添加了基于真实 HF 聊天模板的往返测试，覆盖 Qwen3、MiniMax M2.5、DeepSeek V4 和 GLM-4.7 等模型，并修复了 MiniMax M2 工具解析器对模板空白和流式分割的处理，以及 JSON 数值精度和键顺序保留问题。核心变更包括新增 541 行测试代码和多个核心逻辑修复，已被批准合并。

功能与动机

PR 目的：为 Rust 聊天文本级别添加往返集成覆盖，验证从渲染器、输出处理器到解析器的流水线端到端正确性。修复了两个问题：

- JSON 对象插入顺序和任意精度数值保留，确保 tool-call 参数在模板重新渲染时能精确往返。
- MiniMax M2 解析器对模板空白和流式分割的容忍性。

实现拆解

1. 新增往返测试框架：rust/src/chat/tests/roundtrip.rs 定义了 RoundtripCase 结构体，配置模型 ID、助手停止后缀、解析器选择和 JSON 格式。通过宏生成多个测试函数，针对 reasoning_and_content 和 tool_call_mix 两个 fixture。
2. 修复 MiniMax M2 工具解析器：rust/src/tool-parser/src/minimax_m2.rs 中，将内联的参数解析拆分为独立的 parse_invoke_params 函数，使用 take_until 捕获整个 invoke body 再解析参数，避免流式边界问题。新增 partial_attr_value 函数处理流式分割的属性值。
3. 修复 JSON 数值精度：rust/src/tool-parser/src/parameters.rs 中 convert_number 改为优先使用 serde_json::from_str::<Number> 解析，保留原始数值拼写（如 1.00），降级解析仅作为回退。
4. 添加 trim 辅助：rust/src/chat/src/event.rs 为 AssistantContentBlock 和 AssistantMessage 添加 trim 方法，清理前后空白并移除空块。
5. 修复 tojson filter：rust/src/chat/src/renderer/hf/tojson.rs 中过滤器改为接收 serde_json::Value，并启用 preserve_order 和 arbitrary_precision 特性，保持键顺序和数值拼写。
6. 配置与测试配套：多个 Cargo.toml 添加 paste、serial_test 等依赖并启用特性。取消了一些因依赖 HF 而被忽略的测试（如 Qwen3 生成默认值测试），修正了默认值断言。

rust/src/chat/tests/roundtrip.rs

新增的往返测试文件，是 PR 的核心部分，定义了测试框架和针对 6 个模型的 12 个测试用例。

```
//! Text-level roundtrip tests for the real chat-template and output-processor pairing.
//! The invariant under test is that a structured assistant message rendered as history can be
//! parsed from the generated assistant completion and then rendered back to the exact same
text.

/// One model/parser configuration used to run the fixed roundtrip fixtures.
struct RoundtripCase {
    /// Hugging Face model id resolved through the production backend loader.
    model_id: &'static str,
    /// Final assistant-history suffix rendered by the chat template but not
    /// generated by the model body (consumed by the output processor).
    assistant_stop_suffix: &'static str,
    /// Tool parser selection used by the output processor.
    tool_call_parser: ParserSelection,
    /// Reasoning parser selection used by the output processor.
    reasoning_parser: ParserSelection,
    /// JSON formatting expected after this model's template has materialized tool-call
    arguments.
    json_fmt: JsonFmt,
}

impl RoundtripCase {
    /// Qwen3 XML tool-call format with `qwen3` reasoning tags.
    fn qwen3() -> Self {
        Self {
            model_id: "Qwen/Qwen3-0.6B",
            assistant_stop_suffix: "<|im_end|>\n",
            tool_call_parser: ParserSelection::Auto,
            reasoning_parser: ParserSelection::Auto,
            json_fmt: spaced_json_fmt(), // Qwen3 uses spaced JSON
        }
    }
    // Similar cases for qwen35, minimax_m25, deepseek_v4, glm47, kimi_k25
}

// Macro to generate test functions for each (case, fixture) pair.
macro_rules! roundtrip_tests {
    ($($case:ident => [$($fixture:ident),* $(,)?]),+ $(,)?) => {
        paste::paste! {
            $(
                $(
                    #[tokio::test]
                    #[file_serial([<hf_ $case>])]
                    async fn [<roundtrip_ $case _ $fixture>]() -> Result<()> {
                        [<run_roundtrip_ $fixture>](RoundtripCase::$case()).await
                    }
                )*
            )+
        }
```

```
    };
}

roundtrip_tests! {
    qwen3 => [reasoning_and_content, tool_call_mix],
    qwen35 => [reasoning_and_content, tool_call_mix],
    minimax_m25 => [reasoning_and_content, tool_call_mix],
    deepseek_v4 => [reasoning_and_content, tool_call_mix],
    glm47 => [reasoning_and_content, tool_call_mix],
}
```

rust/src/tool-parser/src/minimax_m2.rs

修复了 MiniMax M2 解析器以处理模板空白和流式分割的参数，核心逻辑变更。

```
/// Parse a complete MiniMax M2 invoke block.
fn invoke_event(input: &mut MinimaxM2Input<'_>) -> ModalResult<MinimaxM2Event> {
    let (name, body) = seq!(
        _: ws0,
        _: literal(INVOKE_START),
        _: (ws1, literal("name=")),
        partial_attr_value, // Use partial-aware version for streaming
        _: literal(">"),
        take_until(0.., INVOKE_END), // Take entire body for later parsing
        _: literal(INVOKE_END),
    )
    .parse_next(input)?;
    // Parse parameters from the complete body, handling whitespace around params
    let raw_params = parse_invoke_params(body)?;

    Ok(MinimaxM2Event::Invoke {
        name: name.trim().to_string(),
        raw_params,
    })
}

/// Parse all parameter blocks inside a complete MiniMax M2 invoke body.
fn parse_invoke_params(invoke_body: &str) -> ModalResult<Vec<(String, String)>> {
    let mut input = invoke_body;
    // Use `eof` to ensure we consume all input, tolerant of whitespace
    delimited(ws0, repeat(0.., terminated(parameter, ws0)), eof)
        .parse_next(&mut input)
}

/// Parse a MiniMax M2 parameter block (on a non-partial &str, so take_until works on complete
slice).
fn parameter(input: &mut &str) -> ModalResult<(String, String)> {
    let (name, value) = seq!(
        _: literal(PARAMETER_START),
        _: (ws1, literal("name=")),
```

```
        attr_value,
        _: literal(">"),
        take_until(0.., PARAMETER_END).map(xml_unescape),
        _: literal(PARAMETER_END),
    )
    .parse_next(input)?;
    Ok((name.trim().to_string(), value.into_owned()))
}

/// Parse a quoted or unquoted XML attribute value from partial streaming input.
fn partial_attr_value<'i>(input: &mut MinimaxM2Input<'i>) -> ModalResult<&'i str> {
    alt((
        delimited(literal("\""), take_until(1.., "\""), literal("\"")),
        delimited(literal("'"), take_until(1.., "'"), literal("'")),
        take_until(1.., ">"), // unquoted attribute ends at '>'
    ))
    .parse_next(input)
}
```

rust/src/tool-parser/src/parameters.rs

修复了 JSON 数值转换以保留原始拼写，例如 1.00 保持为 1.00 而非 1.0。

```
/// Convert one raw string value to a JSON number.
fn convert_number(value: &str) -> Option<Value> {
    // First attempt: parse as serde_json::Number to preserve JSON number spelling
    // (e.g., "5.00" stays as "5.00", "1e0" becomes "1e+0")
    serde_json::from_str::<Number>(value)
        // Fallback to i64 for legacy compatibility (e.g., "+1" -> 1)
        .or_else(|_| value.parse::<i64>().map(Number::from))
        // Final fallback to f64 (e.g., large numbers with decimals that fit in f64)
        .or_else(|_| value.parse::<f64>().ok().and_then(Number::from_f64).ok_or(()))
        .ok()
        .map(Value::Number)
}

#[test]
fn number_conversion_preserves_json_number_spelling_with_legacy_fallback() {
    let params = ToolSchema::from_schema(&json!({
        "type": "object",
        "properties": {
            "value": { "type": "number" }
        }
    }));

    assert_eq!(converted_number_text(&params, "5"), "5");
    assert_eq!(converted_number_text(&params, "5.0"), "5.0");
    assert_eq!(converted_number_text(&params, "5.00"), "5.00"); // Preserved!
    assert_eq!(converted_number_text(&params, "1e0"), "1e+0");
    assert_eq!(converted_number_text(&params, "5."), "5.0");
```

```
    assert_eq!(converted_number_text(&params, "+1"), "1"); // Legacy fallback
    assert_eq!(converted_number_text(&params, "9223372036854775807.5"),
    "9223372036854775807.5"); // Large value preserved
}
```

评论区精华

没有收到人工审核者评论。仅 gemini-code-assist[bot] 自动概述变更，无具体反馈。最终被 njhill 批准。

风险与影响

- 解析器行为变更：MiniMax 解析器从内联改为两步解析，若 body 格式不符合预期可能导致解析失败，但已有新测试覆盖。
- 数值精度兼容性：convert_number 优先保留原始拼写，可能影响依赖旧有规范化行为的上游代码，但测试已覆盖主要场景。
- 新增网络依赖：往返测试依赖 HF 模型访问，但使用 file_serial 避免并发，不会影响 CI 稳定性。
- 测试覆盖提升：显著降低工具调用和推理标签解析的回归风险，易于扩展至更多模型。

关联脉络

该 PR 是 Rust 聊天前端持续增强的一部分，与近期其他 PR（如 #43543 spec-decode attention 修复、#43162 DeepSeek V4 kernel 融合）共同提升了工具调用和模型支持的健壮性。此外，#42789 和 #42768 的 MoE 重构展示了模块化 oracle 设计，而此 PR 关注端到端测试，两者互补。