

PR #54883 完整报告

vllm-project/vllm

[Rust Frontend] Report reasoning tokens in chat completion usage

合并时间: 2026-09-04 07:47

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54883>

执行摘要

本 PR 为 vLLM 的 Rust 前端实现了 `usage.completion_tokens_details.reasoning_tokens` 字段的报告功能，使其在 chat completion API 中与 OpenAI 规范及 Python 前端对齐。核心变更引入了 `ChatTokenUsage` 类型来承载推理令牌计数，并在整个解析、流式处理和响应生成管道中安全地传递该信息。变更涉及 `vllm-chat` 和 `vllm-server` 两个主要 crate，新增了配套的单元测试和端到端测试。

功能与动机

动机：使客户端能够区分消耗在模型推理（思考）过程与最终生成文本上的 token 数量。这在使用 DeepSeek、Qwen 等支持推理的模型时尤为重要。PR body 明确目标：“report `usage.completion_tokens_details.reasoning_tokens` from the Rust frontend”。

解决方案：基于 PR#54884 提供的 token 归因信息（`DecodedText.attributions`），在解析层识别并累计推理令牌，并将其携带至 API 响应的 `usage` 对象中。

实现拆解

- 核心类型引入 (`ChatTokenUsage`): 在 `rust/src/chat/src/event.rs` 中定义了新的 `ChatTokenUsage` 结构体，它包装了引擎级的 `TokenUsage` 并新增 `reasoning_tokens` 字段。通过实现 `From<TokenUsage>` 和 `Deref`，保证了向后兼容性。
- 解析管道改造：
 - `UnifiedParserState`(`rust/src/chat/src/output/default/unified.rs`) 新增 `reasoning_tokens` 字段用于累计。当处理 `UnifiedParserEvent::Reasoning` 时，累加 `piece.attributions.len()`。
 - `AssistantEvent::TextDelta` 变体新增 `token_count: Option<usize>` 字段，用于携带每个增量的令牌计数。对于推理增量，该值为 `Some(attributions.len())`。
 - 新增 `push_piece_delta` 辅助函数，专门处理带归因的推理片段。
- 事件总线扩展：
 - `ChatEvent::BlockDelta` 新增 `token_count: Option<usize>` 字段，将计数从内部事件透传到结构化事件。
 - `ChatEvent::Done` 的 `usage` 字段类型从 `TokenUsage` 更改为 `ChatTokenUsage`。
- 服务器层聚合与输出：

- ContinuousUsage (rust/src/server/src/routes/openai/utils/usage.rs) 新增 reasoning_tokens 字段和累加 / 设置方法，用于在流式响应中生成 continuous_usage_stats。
- 流处理主循环 (rust/src/chat/src/output/default/unified.rs) 在收到 BlockDelta 且 kind 为 Reasoning 时，将 token_count 累加到 ContinuousUsage。收到 Done 事件时，将流式累计值收敛为 Done 事件中的权威值。
- Usage 类型 (rust/src/server/src/routes/openai/utils/types.rs) 新增 completion_tokens_details: CompletionTokenUsageInfo 字段，其 from_token_usage 方法现在接受 ChatTokenUsage。

5. 测试配套：在 vllm-chat 和 vllm-server 的 Rust 单元测试中新增多个用例（如 chunk_stream_reports_reasoning_tokens_with_continuous_usage）。在 Python 端到端测试 test_reasoning_tokens_in_usage 中新增断言。

rust/src/chat/src/output/default/unified.rs

核心解析逻辑实现，负责从 `DecodedText.attributions` 中累计推理令牌并携带到 `AssistantEvent`。

```
// UnifiedParserState 新增字段用于累计最终推理令牌数。
struct UnifiedParserState {
    parser: Box<dyn UnifiedParser>,
    parser_failed: bool,
    open_call_index: Option<usize>,
    /// 仅用于在 Done 事件时生成最终计数；每个 delta 的计数由其自身携带。
    reasoning_tokens: usize,
}

// 在处理解析器输出时，对推理片段进行计数和事件生成。
fn process_parser_output(&mut self, output: UnifiedParserOutput, events: &mut
Vec<AssistantEvent>) -> Result<> {
    for event in output.events {
        match event {
            // ...
            UnifiedParserEvent::Reasoning(piece) => {
                self.open_call_index = None;
                // piece.attributions 恰好包含客户端作为推理接收的所有令牌。
                self.reasoning_tokens += piece.attributions.len();
                push_piece_delta(events, AssistantBlockKind::Reasoning, piece);
            }
            // ...
        }
    }
    Ok(())
}
```

/// 将带归因的 delta 推入事件列表，若文本非空。

```
fn push_piece_delta(events: &mut Vec<AssistantEvent>, kind: AssistantBlockKind, piece:
DecodedText) {
```

```

    if piece.text.is_empty() {
        return;
    }
    events.push(AssistantEvent::TextDelta {
        kind,
        delta: piece.text,
        token_count: Some(piece.attributions.len()), // 此处携带令牌计数
    });
}

```

rust/src/chat/src/event.rs

定义了扩展后的核心数据类型 `ChatTokenUsage` 和事件 `ChatEvent`，是管道中数据流动的契约。

```

/// 一次完成请求的聊天级令牌使用量。
/// 扩展了引擎级的 [TokenUsage]，增加了聊天输出管道测量的推理归属。
#[derive(Debug, Clone, Copy, Default, PartialEq, Eq)]
pub struct ChatTokenUsage {
    /// 引擎级令牌使用量。
    pub engine: TokenUsage,
    /// 归属于推理的生成令牌数；当配置的解析器没有推理通道时为 0。
    pub reasoning_tokens: usize,
}

// 实现便捷转换和解引用。
impl From<TokenUsage> for ChatTokenUsage { /* ... */ }
impl Deref for ChatTokenUsage { /* ... */ }

/// 流式聊天事件。
pub enum ChatEvent {
    // ...
    /// 新观察到的、针对一个打开的助手输出块的增量。
    BlockDelta {
        index: usize,
        kind: AssistantBlockKind,
        delta: String,
        /// 此增量关联的生成令牌数，当可用时。
        token_count: Option<usize>, // 新增字段
    },
    // ...
    /// 终端事件，携带最终的助手消息和完成元数据。
    Done {
        message: AssistantMessage,
        usage: ChatTokenUsage, // 从 TokenUsage 变为 ChatTokenUsage
        // ...
    },
}

```

rust/src/server/src/routes/openai/chat_completions.rs

服务器端流处理主循环，负责聚合连续使用量并生成最终的 API 响应，是功能实现的入口点和关键适配层。

```
// 流式处理循环中，对 BlockDelta 事件进行推理令牌聚合。
Ok(ChatEvent::BlockDelta {
    kind,
    delta,
    token_count,
    ..
}) => {
    // 该计数遵循解码器时钟；即使推理增量对客户端隐藏也提供。
    if matches!(kind, AssistantBlockKind::Reasoning) && let Some(token_count) = token_count {
        continuous_usage.add_reasoning_tokens(token_count);
    }
    // ... 处理 delta 的包含逻辑不变 ...
}
// ...
// 收到 Done 事件时，将流式累计值收敛为权威的终端值。
Ok(ChatEvent::Done {
    usage: final_usage,
    finish_reason,
    ..
}) => {
    // ...
    continuous_usage.set_final_counts(
        final_usage.prompt_token_count,
        final_usage.output_token_count,
    );
    // 收敛流式推理令牌计数到权威的终端值。
    continuous_usage.set_reasoning_tokens(final_usage.reasoning_tokens);
    // ... 后续发送 usage chunk ...
}
```

评论区精华

- 关于 `token_count` 类型：njhill 建议将 `token_count` 改为 `Option<usize>` 以更好区分“已知为零”和“未测量”。作者 BugenZhao 接受了该建议。
- 关于 Harmony 路径：coderabbitai 指出 Harmony 输出路径中 `token_count` 被硬编码为 0，导致推理令牌计数始终为零。作者在代码中添加了 `// TODO: measure reasoning tokens from native analysis-channel token counts`。表示这是已知限制。

风险与影响

1. 类型契约变更：ChatEvent::Done.usage 字段类型从 TokenUsage 变为 ChatTokenUsage。虽然 ChatTokenUsage 通过 Deref 实现暴露了 TokenUsage 的所有字段，但其结构体身份的变化仍可能影响直接模式匹配或序列化下游。测试已覆盖。
2. 数据流正确性：ContinuousUsage 的聚合逻辑需要在推理被隐藏（`include_reasoning=false`）、parser 失败等边界条件下保持正确。PR 中的测试（如

`chunk_stream_reports_reasoning_tokens_when_reasoning_hidden`) 覆盖了这些场景。

3. Harmony 输出缺口：Harmony 输出路径目前未集成推理令牌计数，该部分响应将始终报告 `reasoning_tokens: 0`。此为已知局限性，需后续 PR 解决。

关联脉络

本 PR 是 Rust 前端功能对齐和 token 归因特性链中的一环。它直接依赖 PR#54884 (token 归因信息贯穿解析器层)，并与 PR#54982 (Python 前端的类似功能) 在目标上呼应。近期仓库中 Rust 前端和 token 处理 (如 PR#54896、PR#54845) 有密集活动，表明这是一个活跃的开发方向。