

PR #47289 完整报告

vllm-project/vllm

[Rust Frontend] Recover buffered text from incomplete tool calls at EOS

合并时间: 2026-07-03 11:45

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/47289>

执行摘要

此 PR 修复了 Rust 前端统一包装器在 Gemma4 工具调用流被截断时, 未能恢复 parser 内缓冲文本的问题。现在, 当工具调用因流结束而未能完整闭合时, 客户端会收到截断的原始文本, 而非空消息。

功能与动机

此 PR 解决了当 Gemma4 工具调用流在闭合标记之前被截断时, `finish()` 方法返回错误并丢失缓冲文本的问题。之前, 客户端只能收到 `Done` 和空 assistant 消息, 现在可以收到截断前的缓冲文本作为普通文本 delta。

实现拆解

1. 核心逻辑修改 (`unified.rs`, `finish()` 方法): 在 `parser.finish()` 返回错误时, 新增调用 `self.parser.reset()` 获取缓存的文本, 并通过 `push_text_delta` 将其作为普通文本 delta 添加到事件列表。
2. 更新现有测试: 将 `unified_stream_finish_error_closes_parser_without_reset_text` 重命名为 `unified_stream_finish_error_recovers_buffered_text`, 并调整断言以验证恢复的文本被正确发出。
3. 新增集成测试: `unified_stream_recovers_incomplete_gemma4_tool_call_at_eos` 使用真实的 `Gemma4UnifiedParser` 模拟不完整工具调用流, 验证端到端恢复逻辑。
4. 调整依赖: 在测试模块中引入 `Tool`、`Gemma4UnifiedParser`、`TestTokenizer` 等新导入。

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

核心变更文件: 修改了 `finish()` 方法从 parser 恢复缓冲文本, 并新增和修改了测试用例。

```
// rust/src/chat/src/output/default/unified.rs
// 在 finish() 的错误处理分支中, 调用 parser.reset()
// 获取因工具调用截断而残留的缓冲文本, 并作为普通文本 delta 发出
fn finish(&mut self) -> Result<Vec<AssistantEvent>> {
    let mut events = Vec::new();

    if self.parser_failed {
        return Ok(events);
    }

    match self.parser.finish() {
```

```

Ok(output) => self.process_parser_output(output, &mut events)?,
Err(error) => {
    warn!(
        error = %error.as_report(),
        "unified parser finish failed; closing open parser state"
    );
    self.parser_failed = true;
    self.open_call_index = None;
    // 关键修复：重置 parser 并恢复缓冲文本，而不是忽略它
    let recovered = self.parser.reset();
    push_text_delta(&mut events, AssistantBlockKind::Text, recovered);
}
}

Ok(events)
}

```

评论区精华

无实质 review 讨论。BugenZhao 调用 [@codex review](#) 后直接批准。

风险与影响

- 风险：低。改动局限在 finish() 的错误处理分支，不影响正常流程。测试覆盖了通用错误和 Gemma4 特定场景。
- 影响：对于使用工具调用的 Gemma4 模型，当流被截断时，客户端现在能接收到截断前的文本，提升用户体验。

关联脉络

此 PR 与仓库历史中其他 Rust 前端改进相关，如 [PR #47435]（Rust 前端调度器日志统计补齐），体现团队在持续优化 Rust 前端的稳定性和用户体验。