

PR #53965 完整报告

vllm-project/vllm

[Bugfix] Preserve parallel HY-V3 calls delivered in one streaming delta

合并时间: 2026-08-27 23:26

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/53965>

执行摘要

- 一句话: 修复 HY-V3 流式解析单 delta 中并行工具调用被丢弃
- 推荐动作: 值得精读, 重点是 `_drain_calls` 如何在破坏单调流式语义的前提下追加剩余完整调用, 以及 `content_delta` 的前置文本处理。可以作为 tool parser 处理多调用同一 delta 的参考实现。

功能与动机

PR body 指出: "HY-V3 streaming silently dropped parallel tool calls when more than one complete call arrived in a single engine delta. The parser completed the first call, retained the remaining text in `_buffer`, and returned. If generation had already ended, no later delta arrived to drain that buffer." 该问题会导致并行工具调用在流式模式下丢失, 只能拿到第一个调用。

实现拆解

1. 调整 `extract_tool_calls_streaming` 的前置文本处理: 当 `delta_text` 中包含 标签时, 不再立刻返回 `DeltaMessage(content=text_parts[0])`, 而是将 `content_delta` 记录下来, 在后续分支中与工具调用 delta 合并或单独返回, 避免因提前 return 而错过 buffer 中已经完整的调用。
2. 新增 `_drain_calls(request)` 方法: 循环扫描 `self._buffer`, 每当发现一个以 `tool_call_start_token` 开头、以 `tool_call_end_token` 结尾的完整调用, 就调用非流式解析器 `_extract_tool_calls` 解析, 并更新 `current_tool_id`、`prev_tool_call_arr`、`streamed_args_for_tool` 等流式状态, 生成带顺序 index 的 `DeltaToolCall` 列表。
3. 在 `_extract_streaming_incremental` 的返回路径合并剩余调用: 当当前调用参数流式输出完毕时, 把 `_drain_calls` 返回的多个 `DeltaToolCall` 追加到同一个 `DeltaMessage.tool_calls` 中, 确保一次性输出同 delta 内的后续并行调用。
4. 测试与配套: 本次没有新增测试文件, 仅依赖现有 `tests/tool_parsers/test_hy_v3_tool_parser.py`, 作者运行结果 14 passed。PR 提供了可复现脚本, 说明修复前后输出差异。

关键文件:

- `vllm/tool_parsers/hy_v3_tool_parser.py` (模块 工具解析; 类别 source; 类型 core-logic; 符号 `_drain_calls`, `extract_tool_calls_streaming`, `_extract_streaming_incremental`): 唯一变更文件, 核心修复 HY-V3 流式解析器对并行完整调用的丢弃问题。

关键符号: `_drain_calls`, `extract_tool_calls_streaming`, `_extract_streaming_incremental`

关键源码片段

[vllm/tool_parsers/hy_v3_tool_parser.py](#)

唯一变更文件, 核心修复 HY-V3 流式解析器对并行完整调用的丢弃问题。

```
def _drain_calls(self, request: ChatCompletionRequest) -> list[DeltaToolCall]:
    # 从 buffer 中循环提取完整的工具调用 (以 <tool_call> 开头、</tool_call> 结尾),
    # 直到没有完整调用为止。这样当一个流式 delta 里到达多个并行调用时,
    # 除了当前正在流式输出的调用, 其余完整调用也能立刻被发出。
    deltas: list[DeltaToolCall] = []
    while (start_idx := self._buffer.find(self.tool_call_start_token)) != -1:
        end_idx = self._buffer.find(self.tool_call_end_token, start_idx)
        if end_idx == -1:
            # 未找到结束标记, 说明剩余内容是不完整调用, 留到后续 delta 继续缓冲。
            break
        end_idx += len(self.tool_call_end_token)
        complete_call = self._buffer[start_idx:end_idx]
        self._buffer = self._buffer[end_idx:]
        # 用非流式解析器提取单个完整调用, 通常返回一个 tool_call。
        for call in self._extract_tool_calls(complete_call, request):
            self.current_tool_id += 1
            arguments = call.function.arguments
            self.prev_tool_call_arr.append(
                {"name": call.function.name, "arguments": json.loads(arguments)}
            )
            self.streamed_args_for_tool.append(arguments)
            deltas.append(
                DeltaToolCall(
                    index=self.current_tool_id,
                    id=make_tool_call_id(),
                    type="function",
                    function=DeltaFunctionCall(
                        name=call.function.name,
                        arguments=arguments,
                    ),
                )
            )
    return deltas
```

评论区精华

Reviewer yewentao256 在 PR 评论中提问: "Will this work for case like below? Let's check ...", 即带前置文本的并行调用场景是否也能正确处理。作者随后 (评论内容未完整展示) 进行了说明, 最终 yewentao256 回复 LGTM 并批准, maintainer bbrowning 也确认后合并。

- 含前置文本的并行调用是否被正确处理 (correctness): 作者进行了回应 (具体内容未完整展示), reviewer 认可并批准合并。

风险与影响

- 风险：风险点： 1) `_drain_calls` 依赖 `_extract_tool_calls` 对完整调用文本的解析，如果模型输出包含畸形调用，可能抛异常导致流式中断（不过有外层 `try-except` 保护）； 2) 修改了 `content_delta` 的返回路径，可能影响前置文本与工具调用的顺序语义； 3) 没有配套新增单元测试，回归只能靠现有 14 个用例间接覆盖，若后续有新的并行调用格式变体可能漏测。
- 影响：影响范围：HY-V3 模型（如 DeepSeek 系）使用流式工具调用时会受益，此前丢失的并行调用现在可以正确返回；改动只涉及 `tool parser` 模块，不影响其他模型或非流式路径；团队需注意维护 `_drain_calls` 与 `_extract_tool_calls` 的解析一致性。
- 风险标记：状态清理逻辑新增，缺少新测试覆盖，可能影响流式返回顺序

关联脉络

- PR #52222 [Bugfix][GPT-OSS] Fix strict tool-call grammar to accept Harmony renders: 同属 `tool-calling` 解析器模块的修复，涉及流式 / 非流式解析一致性问题，可互为参考。