PR #41730 完整报告

vllm-project/vllm

[BUGFIX] Support streamed_args_for_tool in MistralToolParser

合并时间：2026-05-06 01:48

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41730

执行摘要

- 一句话：修复 MistralToolParser 流式 tool call 数组越界
- 推荐动作：值得合入并关注后续是否有类似问题的回归报告。合并后建议有人 cherry-pick 到相关 release 分支。

功能与动机

修复 Issue #33916：在使用 --tool-call-parser=mistral 进行流式 tool call 时，出现 IndexError: list index out of range。根本原因是 MistralToolParser 未正确填充 streamed_args_for_tool 和 prev_tool_call_arr，使得 serving 层在判断 finish_reason 时试图访问不存在的索引。

实现拆解

1. 移除硬编码 hack：在 _extract_tool_calls_streaming 方法中，删除原先无条件设置 self.prev_tool_call_arr = [{"arguments": {}}] 的代码段。该 hack 无法正确对应多个 tool call 的场景。
2. 在 tool call 开始时追加列表项：在 _generate_delta_tool_call 方法中检测到新 tool call 开始时（PARSING_NAME 状态），追加 self.streamed_args_for_tool.append("") 和 self.prev_tool_call_arr.append({})，确保列表长度与 tool call 数量一致。
3. 在解析过程中动态更新：在 _generate_delta_tool_call 的 PARSING_NAME 分支中，当工具名解析完成时，写入 prev_tool_call_arr[current_tool_id]["name"]；在 PARSING_ARGUMENTS 分支中，累计 delta_arguments 到 streamed_args_for_tool[current_tool_id]，并同步更新 prev_tool_call_arr[current_tool_id]["arguments"]。
4. 同步 pre-v11 tokenizer 路径：在 _extract_tool_calls_streaming_pre_v11_tokenizer 方法中，于新 tool call 创建时追加列表项，在名称解析完成后更新 name，并调用新增的 _track_streamed_args_pre_v11 方法（与上述逻辑类似）来处理参数累积。
5. 更新测试断言：在 test_tool_parsers/test_mistral_tool_parser.py 的通用测试辅助函数 _test_extract_tool_calls_streaming 中，新增断言验证 streamed_args_for_tool 和 prev_tool_call_arr 长度与元素一致性，并模拟 serving 层的 unstreamed-args 检查逻辑。同时，在 test_extract_tool_calls_streaming_v11_no_tools 中添加断言确保无 tool call 时列表为空。

关键文件：

- vllm/tool_parsers/mistral_tool_parser.py（模块 工具解析器；类别 source；类型 core-logic；符号 _generate_delta_tool_call, _extract_tool_calls_streaming, _extract_tool_calls_streaming_pre_v11_tokenizer, _track_streamed_args_pre_v11）：核心修复文件，修改了流式 tool call 解析的列表追加和更新逻辑，移除了硬编码 hack。
- tests/tool_parsers/test_mistral_tool_parser.py（模块 工具解析器；类别 test；类型 test-coverage）：新增内部状态一致性断言，覆盖有 tool call 和无 tool call 的验证场景。

关键符号：_generate_delta_tool_call, _extract_tool_calls_streaming, _extract_tool_calls_streaming_pre_v11_tokenizer, _track_streamed_args_pre_v11

关键源码片段

vllm/tool_parsers/mistral_tool_parser.py

核心修复文件，修改了流式 tool call 解析的列表追加和更新逻辑，移除了硬编码 hack。

```
# vllm/tool_parsers/mistral_tool_parser.py # 关键变更片段

def _generate_delta_tool_call(self, delta_text: str) -> list[DeltaToolCall]:
    # ... 前略 ...
    if self.streaming_state not in [
        StreamingState.PARSING_NAME,
        StreamingState.PARSING_ARGUMENTS,
    ] and delta_text.startswith(self.bot_token):
        self.current_tool_id += 1
        # 新增：在每个新 tool call 开始时，追加空记录
        self.streamed_args_for_tool.append("")
        self.prev_tool_call_arr.append({})
        self.streaming_state = StreamingState.PARSING_NAME
        delta_text = delta_text.replace(self.bot_token, "", 1)
    if self.streaming_state == StreamingState.PARSING_NAME:
        # ... 省略 ...
        if "{" in delta_text:
            # 工具名解析完成，更新 prev_tool_call_arr 中的 name
            self.prev_tool_call_arr[self.current_tool_id]["name"] = (
                self.current_tool_name
            )
            self.streaming_state = StreamingState.PARSING_ARGUMENTS
    if self.streaming_state == StreamingState.PARSING_ARGUMENTS:
        # ... 省略 ...
        # 新增：累积参数并同步更新两条列表
        self.streamed_args_for_tool[self.current_tool_id] += delta_arguments
        self.prev_tool_call_arr[self.current_tool_id]["arguments"] = (
            self.streamed_args_for_tool[self.current_tool_id]
        )
        # ... 返回 delta 等后续逻辑 ...
    return []
```

```
# 对应 pre-v11 tokenizer 路径也有类似修改，核心模式一致。
```

tests/tool_parsers/test_mistral_tool_parser.py

新增内部状态一致性断言，覆盖有 tool call 和无 tool call 的验证场景。

```
# tests/tool_parsers/test_mistral_tool_parser.py # 新增断言片段

# 在 _test_extract_tool_calls_streaming 函数结尾新增:
if expected_tool_calls:
    # 验证内部状态列表长度与期望 tool call 数量一致
    assert len(tool_parser.streamed_args_for_tool) == len(expected_tool_calls)
    assert len(tool_parser.prev_tool_call_arr) == len(expected_tool_calls)
    for i in range(len(expected_tool_calls)):
        # prev_tool_call_arr 中的 arguments 应等于 streamed_args_for_tool[i]
        assert (
            tool_parser.prev_tool_call_arr[i]["arguments"]
            == tool_parser.streamed_args_for_tool[i]
        )
        # streamed_args_for_tool[i] 应等于实际累积的参数字符串
        assert tool_parser.streamed_args_for_tool[i] == function_args_strs[i]
        # prev_tool_call_arr 中的 name 应等于期望的工具名
        assert (
            tool_parser.prev_tool_call_arr[i]["name"]
            == expected_tool_calls[i].function.name
        )
    # 模拟 serving 层的 unstreamed-args 检查（剩余 JSON 片段应为空）
    index = len(tool_parser.prev_tool_call_arr) - 1
    args = tool_parser.prev_tool_call_arr[index].get("arguments", {})
    expected_call = (
        args if isinstance(args, str) else json.dumps(args, ensure_ascii=False)
    )
    actual_call = tool_parser.streamed_args_for_tool[index]
    remaining_call = expected_call.replace(actual_call, "", 1)
    assert remaining_call == ""
else:
    # 无 tool call 时两个列表应均为空
    assert len(tool_parser.streamed_args_for_tool) == 0
    assert len(tool_parser.prev_tool_call_arr) == 0
```

评论区精华

无人工审核评论。仅 bot（gemini-code-assist）总结变更内容，sfeng33 直接 approve。

- 暂无高价值评论线程

风险与影响

- 风险：

1. 回归风险（低）：变更集中在两个方法内，移除了一个粗暴的 hack，改为基于状态的动态追加。如果流式解析过程中出现状态机未覆盖的分支，可能导致列表长度不匹配，但现有测试覆盖了主要场景。
2. 兼容性风险（低）：prev_tool_call_arr 的键名由硬编码的 arguments 改为动态更新的 name 和 arguments，serving 层（serving_chat.py）读取这些字段的方式需保持一致。鉴于该 PR 就是针对 serving 层的问题修复，风险可控。
3. 性能影响（低）：每次 tool call 解析增加少量列表操作，对整体吞吐影响可忽略。 - 影响：影响范围：使用 Mistral 模型且开启 tool calling 流式输出的用户。影响程度：修复了 IndexError 崩溃，使流式 tool call 功能恢复正常。无 breaking change。对团队：代码可读性提升，移除了带 "HACK" 注释的临时代码，改为主流的状态管理方式。

- 风险标记：缺少回归测试覆盖多轮 tool call

关联脉络

- 暂无明显关联 PR