PR #40148 完整报告

vllm-project/vllm

fix(openai): tolerate empty content in forced tool choice

合并时间：2026-05-06 22:16

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40148

执行摘要

- 一句话：修复 forced tool_choice 在 content=None 时的断言崩溃
- 推荐动作：值得精读。该 PR 展示了如何正确修复推理解析与工具调用交互的边界情况，并包含 review 中设计决策的来回迭代，对理解工具调用流程有参考价值。

功能与动机

Issue #40147 报告：推理解析器返回 content=None 时，forced tool_choice 触发 AssertionError，导致服务器错误而非有效的空函数调用。需要使服务器能够优雅处理 content 为 None 的情况，符合 OpenAI 行为。

实现拆解

1. 修改 OpenAIServing._parse_tool_calls_from_content（vllm/entrypoints/openai/engine/serving.py）：在 Responses API 的 ToolChoiceFunction 分支和 ChatCompletion 的 ChatCompletionNamedToolChoiceParam 分支中，将 assert content is not None 替换为 if content is None: return []，None，确保 content 为空时不构造 FunctionCall。
2. 修改 DelegatingParser._parse_tool_calls（vllm/parser/abstract_parser.py）：同样将 forced function call 分支的断言改为条件返回空列表。
3. 调整 chat_completion/serving.py 中 tool_calls 的断言：将 assert tool_calls is not None and len(tool_calls) > 0 简化为 tool_calls = tool_calls or []，避免 tool_calls 为 None 时崩溃。
4. 新增单元测试文件 tests/entrypoints/openai/test_tool_choice_content_none.py，覆盖 ChatCompletion 和 Responses 路径中 named tool_choice 传递 content=None 的场景。
5. 扩展端到端测试 test_max_tokens_with_tool_choice_required（tests/entrypoints/openai/chat_completion/test_completion_with_function_calling.py），通过参数化同时测试 'required' 和 named function 两种 tool_choice。

关键文件：

- vllm/entrypoints/openai/engine/serving.py（模块 服务层；类别 source；类型 core-logic；符号 _parse_tool_calls_from_content）：核心修复文件，在 _parse_tool_calls_from_content 中添加 content 为 None 的守卫条件。

- vllm/parser/abstract_parser.py（模块 解析器；类别 source；类型 core-logic；符号 _parse_tool_calls）：在 DelegatingParser._parse_tool_calls 中做相同修复。
- tests/entrypoints/openai/test_tool_choice_content_none.py（模块 测试；类别 test；类型 test-coverage；符号 _DummyDelegatingParser, test_parse_tool_calls_from_content_allows_named_tool_choice_with_none_content, test_responses_parser_allows_named_tool_choice_with_none_content）：新增回归测试，验证 content=None 时返回空工具调用。
- tests/entrypoints/openai/chat_completion/test_completion_with_function_calling.py（模块 函数调用测试；类别 test；类型 test-coverage；符号 test_max_tokens_with_tool_choice_required）：扩展端到端测试，覆盖 named tool_choice 场景。
- vllm/entrypoints/openai/chat_completion/serving.py（模块 服务层；类别 source；类型 core-logic；符号 chat_completion_full_generator）：修复流式响应中 tool_calls 断言过于严格。

关键符号：_parse_tool_calls_from_content, _parse_tool_calls, chat_completion_full_generator

关键源码片段

vllm/entrypoints/openai/engine/serving.py

核心修复文件，在 _parse_tool_calls_from_content 中添加 content 为 None 的守卫条件。

```
# 在 _parse_tool_calls_from_content 方法中处理强制工具调用的两个分支：
# 1. ToolChoiceFunction（Responses API）
# 2. ChatCompletionNamedToolChoiceParam（Chat API）
# 此前使用 assert content is not None 会在此处抛出异常，
# 现在改为守卫条件，当 content 为 None 时直接返回空列表。

# 分支 1: Responses API forced function call
if (
    not use_mistral_tool_parser
    and request.tool_choice
    and isinstance(request.tool_choice, ToolChoiceFunction)
):
    # 当推理解析器（如 DeepSeekV3ReasoningParser）返回 content=None 时，
    # 返回空列表，避免 assert 崩溃
    if content is None:
        return [], None
    function_calls.append(
        FunctionCall(name=request.tool_choice.name, arguments=content)
    )
    content = None # 标记 content 已用于工具调用

# 分支 2: ChatCompletion named tool choice
elif (
    not use_mistral_tool_parser
```

```
        and request.tool_choice
        and isinstance(request.tool_choice, ChatCompletionNamedToolChoiceParam)
        and (tool_parser_cls is None or tool_parser_cls.supports_required_and_named)
    ):
        # 同样，content 为 None 时返回空列表
        if content is None:
            return [], None
        function_calls.append(
            FunctionCall(name=request.tool_choice.function.name, arguments=content)
        )
        content = None
```

tests/entrypoints/openai/test_tool_choice_content_none.py

新增回归测试，验证 content=None 时返回空工具调用。

```
class _DummyDelegatingParser(DelegatingParser):
    """模拟一个始终返回 None 的推理解析器"""
    def extract_reasoning(self, model_output: str, request):
        # 返回 (None, model_output) 触发 content=None 场景
        return None, model_output
    # 其他方法省略 ...

def test_parse_tool_calls_from_content_allows_named_tool_choice_with_none_content():
    """验证 ChatCompletion named tool_choice 在 content=None 时返回空列表"""
    request = ChatCompletionRequest.model_validate({
        "model": "test-model",
        "messages": [{"role": "user", "content": "test"}],
        "tools": [{
            "type": "function",
            "function": {"name": "get_weather", "parameters": {}},
        }],
        "tool_choice": {"type": "function", "function": {"name": "get_weather"}},
    })
    tool_calls, content = OpenAIServing._parse_tool_calls_from_content(
        request=request, tokenizer=None, enable_auto_tools=True,
        tool_parser_cls=None, content=None,
    )
    assert content is None
    assert tool_calls is not None
    assert tool_calls == [] # 现在是空列表，而非崩溃
```

评论区精华

Reviewer chaunceyjiang 最初建议在 DeepSeek 解析器中修复（"Perhaps it would be better to modify the corresponding parser"），但作者 QwertyJack 重写后，reviewer 纠正了自己的看法（"I realized I was wrong earlier"），并指出更正确的修复是当 content 为 None 时返回空工具调用列表（[]），而不是拼接空的参数字符串。最终方案采纳了这一建议，并移除了之前尝试的 content = content or '' 逻辑。讨论还涉及 arguments 应使用 "{}" 还是 "

"，但最终决定直接返回空列表，符合 OpenAI 行为。

- 修复位置：改 parser 还是改入口？ (design): 最终在 shared forced-tool handling 路径中返回空列表，不修改 DeepSeek parser。
- arguments 格式：应使用 "{}" 还是 ""？ (design): 最终决定不合成函数调用，而是返回空列表 []，避免 arguments 格式争议。
- 是否需要合成空的 FunctionCall？ (design): 采纳空列表，与 OpenAI 行为一致。

风险与影响

- 风险：风险极低：改动仅限于 forced tool_choice 路径中 content 为 None 的边界情况，替换断言为条件返回，不会影响其他 tool_choice 模式（auto, required, none）。新增单元测试覆盖 ChatCompletion 和 Responses 两种入口，扩展的端到端测试覆盖了 max_tokens 触发 content 为 None 的场景。唯一潜在风险是如果调用方依赖原先的断言行为（即期望服务器报错），但这是从内部错误恢复为正常响应，更符合预期。
- 影响：对用户：修复了 DeepSeek V3 等带推理的模型在使用 forced tool_choice 时可能返回 500 服务器错误的问题，现在会正确返回空工具调用。对系统：无性能开销，分支判断仅在 content 为 None 时执行。对团队：代码更健壮，减少了类似异常的排查成本。
- 风险标记：边界条件处理，测试覆盖新增

关联脉络

- PR #40147 [Bug]: forced tool_choice asserts when reasoning extraction returns content=None: 直接关联的 issue，描述问题根因和复现步骤。