PR #44105 完整报告

vllm-project/vllm

[BugFix] Omit empty tool_calls from OpenAI chat responses

合并时间：2026-06-23 13:43

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44105

执行摘要

- 一句话：修复空 tool_calls 数组导致 OpenAI SDK 失败
- 推荐动作：该 PR 值得精读，原因如下：
  1. 展示了如何利用 Pydantic V2 的 @model_serializer 优雅地调整序列化输出，同时保持模型定义不变。
  2. 测试覆盖设计良好，既验证了 model_dump 又验证了 model_dump_json，并验证了 OpenAI SDK 模型解析。
  3. 修复了一个真实影响用户的兼容性 bug，展现了良好的用户同理心。
  4. review 中关于条件判断方式的讨论体现了对代码质量的关注。

功能与动机

修复 Issue #44104 中描述的 OpenAI 兼容性 bug：当聊天补全 API 在工具调用结果后返回普通文本响应时，序列化输出中包含空的 tool_calls 数组。OpenAI SDK 客户端将 tool_calls 非 None 视为工具调用路径，从而在访问空列表时索引失败。需要对齐 OpenAI 规范，仅在确实存在工具调用时才包含 tool_calls 字段。

实现拆解

1. 识别问题：在 ChatMessage（非流式）和 DeltaMessage（流式）的 Pydantic 序列化中，默认的 tool_calls 字段由 Field(default_factory=list) 产生空列表，序列化后输出 tool_calls: []，与 OpenAI 规范不一致。
2. 添加序列化钩子：分别在 vllm/entrypoints/openai/chat_completion/protocol.py 的 ChatMessage 类和 vllm/entrypoints/openai/engine/protocol.py 的 DeltaMessage 类上，使用 @model_serializer(mode="wrap") 装饰器添加 _serialize 方法。该方法先调用默认序列化 handler，然后检查 data.get("tool_calls", []) 是否为空，若是则移除该键。使用 len(...) == 0 而非直接比较 == [] 以提高稳健性。
3. 更新现有测试：修改 test_completion_with_function_calling.py 和 test_serving_chat.py 中的断言，将 assert len(choice.message.tool_calls) == 0 改为 assert choice.message.tool_calls is None，以反映新行为。
4. 新增单元测试：在 test_tool_choice_content_none.py 中新增辅助函数 _chat_response 和 _stream_response，以及 4 个测试函数，验证 JSON payload 中 tool_calls 字段在空时被省略、非空时保留，并验证通过 OpenAI Python SDK 模型解析时 tool_calls 为 None。

5. 验证无回归：通过 git grep 确认 vLLM 内部代码无直接依赖 message["tool_calls"] 键；运行所有相关测试通过，ruff 检查和 pytest 通过。

关键文件：

- vllm/entrypoints/openai/chat_completion/protocol.py（模块 聊天补全；类别 source；类型 core-logic；符号 _serialize）：核心变更点：为 ChatMessage 类添加 _serialize 序列化方法，当 tool_calls 为空时从输出中移除
- vllm/entrypoints/openai/engine/protocol.py（模块 引擎协议；类别 source；类型 core-logic；符号 _serialize）：核心变更点：为 DeltaMessage 类添加 _serialize 序列化方法，使流式响应中的空 tool_calls 被省略
- tests/entrypoints/openai/test_tool_choice_content_none.py（模块 工具测试；类别 test；类型 test-coverage；符号 _chat_response, test_chat_completion_response_omits_empty_tool_calls_payload, test_chat_completion_response_keeps_non_empty_tool_calls_payload, _stream_response）：新增大量单元测试，覆盖非流式和流式两种模式下的空 / 非空 tool_calls 场景，是验证修复正确性的核心测试文件
- tests/entrypoints/openai/chat_completion/test_completion_with_function_calling.py（模块 函数测试；类别 test；类型 test-coverage；符号 test_max_tokens_with_tool_choice_required）：调整现有测试断言以匹配新行为：当 finish_reason=length 且 tool_calls 为空时，现在期望 tool_calls 为 None 而非空列表
- tests/entrypoints/openai/chat_completion/test_serving_chat.py（模块 服务测试；类别 test；类型 test-coverage；符号 test_gpt_oss_tool_choice_none）：调整现有测试断言：test_gpt_oss_tool_choice_none 中期望 tool_calls 为 None

关键符号：ChatMessage._serialize, DeltaMessage._serialize

关键源码片段

vllm/entrypoints/openai/chat_completion/protocol.py

核心变更点：为 ChatMessage 类添加 _serialize 序列化方法，当 tool_calls 为空时从输出中移除

```
# vllm/entrypoints/openai/chat_completion/protocol.py
class ChatMessage(OpenAIBaseModel):
    role: str
    content: str | None = None
    refusal: str | None = None
    # ... 其他字段 ...
    tool_calls: list[ToolCall] = Field(default_factory=list)

    # 添加序列化钩子，在序列化输出时若 tool_calls 为 [] 则移除该字段
    @model_serializer(mode="wrap")
    def _serialize(self, handler):
        data = handler(self)
        # 当 tool_calls 为空列表时，从 dict 中弹出该键，使输出中不包含该字段
        # 使用 len() 检查而非直接比较 == []，提高代码健壮性
        if len(data.get("tool_calls", [])) == 0:
```

```
            data.pop("tool_calls", None)
        return data
```

vllm/entrypoints/openai/engine/protocol.py

核心变更点：为 DeltaMessage 类添加 _serialize 序列化方法，使流式响应中的空 tool_calls 被省略

```
# vllm/entrypoints/openai/engine/protocol.py
class DeltaMessage(OpenAIBaseModel):
    role: str | None = None
    content: str | None = None
    reasoning: str | None = None
    tool_calls: list[DeltaToolCall] = Field(default_factory=list)

    # 序列化时若 tool_calls 为 [] 则移除，与 ChatMessage 行为一致
    @model_serializer(mode="wrap")
    def _serialize(self, handler):
        data = handler(self)
        if len(data.get("tool_calls", [])) == 0:
            data.pop("tool_calls", None)
        return data
```

tests/entrypoints/openai/test_tool_choice_content_none.py

新增大量单元测试，覆盖非流式和流式两种模式下的空 / 非空 tool_calls 场景，是验证修复正确性的核心测试文件

```
# tests/entrypoints/openai/test_tool_choice_content_none.py

# 辅助函数：构造非流式响应
def _chat_response(message: ChatMessage) -> ChatCompletionResponse:
    return ChatCompletionResponse(
        model="test-model",
        choices=[ChatCompletionResponseChoice(
            index=0, message=message, finish_reason="stop")],
        usage=UsageInfo(prompt_tokens=1, completion_tokens=1, total_tokens=2),
    )

# 测试空 tool_calls 被省略
def test_chat_completion_response_omits_empty_tool_calls_payload():
    # 构造只有 content 的 ChatMessage（tool_calls 使用默认空列表）
    response = _chat_response(ChatMessage(role="assistant", content="done"))
    payload = response.model_dump()
    # 验证序列化后 choices[0].message 中不包含 tool_calls 键
    assert "tool_calls" not in payload["choices"][0]["message"]
    # 同时验证 exclude_unset=True 模式下也省略
    payload_exclude_unset = response.model_dump(exclude_unset=True)
    assert "tool_calls" not in payload_exclude_unset["choices"][0]["message"]
    # 使用 OpenAI SDK 模型解析，验证 tool_calls 属性为 None
    parsed = OpenAIChatCompletion.model_validate(payload)
```

```
    assert parsed.choices[0].message.tool_calls is None

# 测试非空 tool_calls 被保留
def test_chat_completion_response_keeps_non_empty_tool_calls_payload():
    # 构造包含一个 tool_call 的 ChatMessage
    response = _chat_response(ChatMessage(
        role="assistant", content="",
        tool_calls=[ToolCall(
            function=FunctionCall(name="get_weather",
                                  arguments='{"city": "Beijing"}'))]))
    message = response.model_dump()["choices"][0]["message"]
    assert len(message["tool_calls"]) == 1
    assert message["tool_calls"][0]["function"]["name"] == "get_weather"
```

评论区精华

- Reviewer Chauncey Jiang在审查时建议使用 len(data.get("tool_calls", [])) == 0 替代 data.get("tool_calls") == []，以提高代码清晰度和避免潜在的空列表比较问题。作者采纳了该建议。
- Issue 评论者 hclsys指出该变更可能影响内部通过 message["tool_calls"] 读取空数组的消费者（如回调、日志）。作者通过 git grep 检查确认 vLLM 内部无此类依赖，风险可控。
- 序列化条件判断改进 (style): 作者采纳建议，在最终代码中使用 len(...) == 0。
- 对内部消费者的影响 (question): 确认无内部消费者依赖，无需额外处理。

风险与影响

- 风险：主要风险包括：
  - 兼容性：对于依赖 tool_calls 键存在的下游工具或自定义中间件，省略该键可能导致 KeyError。但经过 grep 内部代码，未发现 vLLM 内部有此类依赖；外部用户通常使用 OpenAI SDK，该 SDK 会处理 None 字段。
  - 回归风险：改动集中在两个类的方法，每个方法只有 7 行，测试新增 4 个单元测试并更新 2 个现有测试，覆盖空和非空场景，回归概率低。
  - 序列化性能：model_serializer 模式会增加微小的函数调用开销，但影响可忽略。
  - 多格式兼容：变更同时影响 model_dump() 和 model_dump_json()，行为一致。未修改 model_dump(by_alias=True) 等特殊模式，但当前字段无别名，无影响。
  - 影响：
  - 用户影响：使用 OpenAI SDK 或其他基于 OpenAI 规范解析响应的客户端将不再因空 tool_calls 而崩溃。这是正向影响。
  - 系统影响：序列化输出格式变化，但非空 tool_calls 行为不变。API 响应体稍微变小（少一个字段）。需确保监控和日志系统不会因缺少字段而出错（但日志通常序列化后记录，不会受影响）。
  - 团队影响：无直接运维影响，更新部署后即可。
  - 风险标记：序列化行为变更

关联脉络

- 暂无明显关联 PR