PR #42189 完整报告

vllm-project/vllm

[Bugfix] [Frontend] Responses API, fix merging of messages

合并时间：2026-05-13 00:10

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42189

执行摘要

- 一句话：修复 Responses API 消息合并，合并连续 assistant 消息
- 推荐动作：该 PR 值得精读，尤其是学习如何通过函数签名设计（prev_msg 参数）将隐式的合并策略直接集成到构造逻辑中，替代独立的合并函数。review 中关于类型安全的讨论也值得参考。

功能与动机

Issue #37167 指出 Responses API 中消息与工具调用未正确合并，导致生成连续 assistant 消息，违反 Chat 模板要求。PR 目标是合并连续的 assistant 端 Items（output_text, function_call, reasoning）为单个 assistant 消息，保持单轮 assistant 交互。

实现拆解

实现分为三步：

1. 替换入口函数：construct_chat_messages_with_tool_call 从包裹 _construct_single_message_from_response_item + _maybe_combine_reasoning_and_tool_call 的配对改为单一调用 _construct_message_from_response_item，直接传入前一消息用于合并。
2. 重构核心逻辑：新函数 _construct_message_from_response_item 接收 prev_msg，根据当前 Item 类型和前一消息状态决定合并或新建消息。 - ResponseFunctionToolCall：若前一消息是 assistant 且无 tool_calls，将新工具调用追加到其 tool_calls 列表，返回 None（原消息被更新）。 - ResponseOutputMessage：若前一消息是 assistant 且无 content，追加内容；否则新建。 - ResponseReasoningItem：类似处理，合并 reasoning 字段。
3. 测试覆盖：扩展测试文件，添加帮助函数 make_output_message、make_reasoning_item、make_function_call、make_function_call_output 和 _single_chat_message；新增 test_construct_chat_messages_preserves_single_item_conversions 测试各种组合的合并行为。

关键文件：

- vllm/entrypoints/openai/responses/utils.py（模块 响应工具；类别 source；类型 core-logic；符号 _construct_message_from_response_item, construct_chat_messages_with_tool_call, _maybe_combine_reasoning_and_tool_call, _construct_single_message_from_response_item）：重构消息合并逻辑：将

split-and-combine 模式替换为带状态合并的单一函数 _construct_message_from_response_item。

- tests/entrypoints/openai/responses/test_responses_utils.py（模块 测试；类别 test；类型 test-coverage；符号 _single_chat_message, make_output_message, make_reasoning_item, make_function_call）：扩展测试套件，添加帮助函数和新的合并行为测试，覆盖 output_text、reasoning、function_call 的各种合并场景。

关键符号：_construct_message_from_response_item, construct_chat_messages_with_tool_call, _single_chat_message, make_function_call

关键源码片段

tests/entrypoints/openai/responses/test_responses_utils.py

扩展测试套件，添加帮助函数和新的合并行为测试，覆盖 output_text、reasoning、function_call 的各种合并场景。

测试帮助函数

```
def _single_chat_message(item):
    # 从单个 item 构造消息，确保结果非空
    message = _construct_message_from_response_item(item)
    assert message is not None, 'Expected a message for item'
    return message


def make_function_call(
    *, call_id: str, name: str = 'test_function',
    arguments: str = '{}', id: str = 'tool_id',
    status: str | None = None,
) -> ResponseFunctionToolCall:
    kwargs = {
        'type': 'function_call',
        'id': id,
        'call_id': call_id,
        'name': name,
        'arguments': arguments,
    }
    if status is not None:
        kwargs['status'] = status
    return ResponseFunctionToolCall(**kwargs)
```

合并行为测试

```
def test_construct_chat_messages_preserves_single_item_conversions():
    # 验证 input_text item 保持转换为 assistant content
    msg_out = make_output_message('Hello')
    result = construct_chat_messages_with_tool_call([msg_out])
    assert len(result) == 1
    assert result[0]['role'] == 'assistant'
```

```
    assert result[0]['content'] == 'Hello'

    # 验证 reasoning item 转换为 reasoning 字段
    reasoning = make_reasoning_item(content_text='Thinking...')
    result = construct_chat_messages_with_tool_call([reasoning])
    assert len(result) == 1
    assert result[0]['reasoning'] == 'Thinking...'

    # 验证 tool call 转换为 tool_calls
    tool = make_function_call(call_id='call_1')
    result = construct_chat_messages_with_tool_call([tool])
    assert len(result) == 1
    assert result[0]['tool_calls'][0]['id'] == 'call_1'

    # 验证 reasoning + tool call 合并为一条消息
    result = construct_chat_messages_with_tool_call([reasoning, tool])
    assert len(result) == 1
    assert result[0]['reasoning'] == 'Thinking...'
    assert result[0]['tool_calls'][0]['id'] == 'call_1'
```

评论区精华

- 合并范围讨论：gemini-code-assist 指出新实现只合并 input_messages 内部，不与历史消息（prev_msg/prev_response_output）合并。作者回应历史消息已在 Chat 模板格式中，无需再次合并，prev_response_output 超出当前范围。
- 类型安全性讨论：gemini-code-assist 指出使用 assert 确保 tool_calls 是列表不可靠，且可能为不可变 tuple 导致 append 失败。作者修复：移除 assert，将 tool_calls 转换为列表后再操作。
- 日志移除：chaunceyjiang 建议移除 reasoning 跳过时的 warning 日志，作者同意并删除。
  - 合并范围仅限 input_messages，未与历史 assistant 消息合并 (design): 作者回应 prev_msg 已在 Chat 模板格式中无需合并，prev_response_output 超出当前范围，不做处理。
  - 使用 assert 进行类型验证的风险及 tool_calls 不可变序列问题 (correctness): 作者移除 assert，添加 isinstance 检查并转换为 list。
  - 移除多余的 warning 日志 (style): 作者同意并删除。

风险与影响

- 风险：
  - 合并策略变更可能影响上游：依赖于旧行为（总是新建 assistant 消息）的调用方可能因合并导致消息结构不同，但根据规范合并是正确行为。
  - 历史消息未合并：当前合并仅覆盖当前输入内的 items，不处理与历史输出的合并，可能仍在极端场景下出现连续 assistant 消息，但作者声明此设计。
  - 测试覆盖虽然增加，但仍缺少与历史消息交互的端到端测试。
- 影响：

- 用户影响：使用 Responses API 进行多轮工具调用和推理的模型（如 Qwen3、Gemma4）将生成正确的 Chat 格式消息，修复此前导致模板错误或推理崩溃的问题。BFCL 评估显示多轮准确率显著提升（例如 Gemma4 multi_turn_base 从 28.5% 提升至 79.5%）。
- 系统影响：仅修改前端工具函数，无性能或延迟引入。
- 团队成员影响：简化了消息构造逻辑，后续维护成本降低。
- 风险标记：核心路径变更，兼容性影响，历史消息合并 gap

关联脉络

- PR #37294 [Responses] Combine msg (from PR body reference): 此 PR 基于 #37294 的思路重构，解决了相同的消息合并问题。