PR #40700 完整报告

vllm-project/vllm

[Frontend]Responses API supports Tool/Function calling with streaming with required

合并时间：2026-04-28 10:36

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40700

执行摘要

- 一句话：Responses API 新增流式工具调用支持 required tool_choice
- 推荐动作：此 PR 值得精读，尤其是 vllm/tool_parsers/streaming.py 的状态机设计和 vllm/entrypoints/openai/responses/streaming_events.py 的事件发射模式。设计决策包括将公共流式解析逻辑抽取为独立函数以促进复用，以及使用 SimpleStreamingState 显式管理流式状态而不是依赖临时变量。需要注意的是，review 中提出的 history_tool_call_cnt 未递增和硬编码路径等问题尚未解决，在合并后可能需要后续 PR 跟进。建议阅读时同时关注这些评论，以全面理解潜在问题。

功能与动机

Responses API 是 vLLM 提供的 OpenAI Responses API 兼容接口，此前在流式（streaming）模式下无法处理 tool_choice 为 required 或 named 的工具调用请求，导致该场景下功能缺失。此 PR 旨在填补这一空白，使 Responses API 的流式行为更符合 OpenAI API 规范，同时通过代码抽取提高可维护性。

实现拆解

1. 抽取公共工具调用流式解析函数（vllm/tool_parsers/streaming.py）：新建 streaming.py 模块，包含 _bracket_level、filter_delta_text、extract_named_tool_call_streaming 和 extract_required_tool_call_streaming 等函数。这些函数原为 OpenAIServingChat 类的私有方法，现成为独立函数，供聊天补全和 Responses API 共享使用。
2. 增强 Responses API 流式事件生成（vllm/entrypoints/openai/responses/streaming_events.py）：引入 _StateType 枚举和 SimpleStreamingState 数据类，将流式状态分解为 NONE、CONTENT、REASONING、TOOL_CALL 四种状态。新增 emit_simple_content_open、emit_simple_content_delta、emit_simple_content_done、emit_simple_reasoning_open、emit_simple_reasoning_delta、emit_simple_reasoning_done 以及工具调用相关发射函数，能够根据当前状态生成正确的 SSE 事件流。
3. 简化 Responses 服务端流处理（vllm/entrypoints/openai/responses/serving.py）：将 _process_simple_streaming_events 方法重写，使用 SimpleStreamingEventProcessor 替代原有的临时变量和条件分支，大幅减少了代码量（-573 行），提高了可读性和可维护性。同时新增 _get_logprobs 辅助方法。

4. Parser 层增加 required/named 支持（vllm/parser/abstract_parser.py）：在 Parser 类中添加 _extract_tool_calls_streaming 方法，根据 tool_choice 类型分派到 extract_required_tool_call_streaming 或原有的 extract_tool_calls_streaming。在 StreamState 中添加 history_tool_call_cnt、tool_call_id_type、function_name_returned 字段以支持流式状态跟踪。parse_delta 方法被修改为使用新的 _extract_tool_calls_streaming 并更新状态。
5. 清理聊天补全重复代码（vllm/entrypoints/openai/chat_completion/serving.py）：删除 _bracket_level 和 _filter_delta_text 私有方法以及 extract_tool_call_required_streaming 的内联实现，改为从 vllm.tool_parsers.streaming 导入 extract_required_tool_call_streaming 函数，简化了代码。
6. 新增集成测试（tests/entrypoints/openai/responses/test_function_call.py）：在现有测试基础上，添加 test_function_calling_with_streaming_forced_tool_choice 测试用例，对 tool_choice='required' 和 'auto' 进行参数化测试，验证流式事件序列的正确性。

关键文件：

- vllm/tool_parsers/streaming.py（模块 工具解析；类别 source；类型 core-logic；符号 _bracket_level, filter_delta_text, extract_named_tool_call_streaming, extract_required_tool_call_streaming）：新增流式工具调用解析模块，集中了 filter_delta_text、extract_required_tool_call_streaming 等核心函数，是此 PR 的核心抽象
- vllm/entrypoints/openai/responses/streaming_events.py（模块 流式事件；类别 source；类型 core-logic；符号 _StateType, SimpleStreamingState, emit_simple_content_open, emit_simple_content_delta）：新增 SimpleStreamingState 状态机和事件发射函数，是 Responses API 流式工具调用的事件生成核心
- vllm/entrypoints/openai/chat_completion/serving.py（模块 聊天补全；类别 source；类型 dependency-wiring；符号 _bracket_level, _filter_delta_text）：删除冗余的私有方法并导入共享的 streaming 模块，简化聊天补全流式工具调用处理
- vllm/entrypoints/openai/responses/serving.py（模块 响应服务；类别 source；类型 core-logic；符号 _get_logprobs）：重写 _process_simple_streaming_events 使用新的事件处理器，大幅减少代码量并提高可维护性
- vllm/parser/abstract_parser.py（模块 解析器；类别 source；类型 core-logic；符号 _extract_tool_calls_streaming）：增加 _extract_tool_calls_streaming 方法以支持 required/named tool choice 流式解析，并维护 StreamState 中的计数
- tests/entrypoints/openai/responses/test_function_call.py（模块 功能测试；类别 test；类型 test-coverage；符号 test_function_calling_with_streaming_forced_tool_choice）：添加对 required 和 auto tool_choice 的流式测试，验证 SSE 事件序列

关键符号：_bracket_level, filter_delta_text, extract_named_tool_call_streaming, extract_required_tool_call_streaming, SimpleStreamingState, emit_simple_content_open, emit_simple_content_delta, emit_simple_content_done, emit_simple_reasoning_open, emit_simple_reasoning_delta, emit_simple_reasoning_done, _extract_tool_calls_streaming, _process_simple_streaming_events

关键源码片段

vllm/tool_parsers/streaming.py

新增流式工具调用解析模块，集中了 filter_delta_text、extract_required_tool_call_streaming 等核心函数，是此 PR 的核心抽象

```
# vllm/tool_parsers/streaming.py
# 计算字符串中的花括号嵌套层级
def _bracket_level(s: str, opening: str = "{", closing: str = "}") -> int:
    level = 0
    for char in s:
        if char == opening:
            level += 1
        elif char == closing:
            level -= 1
    return level


# 过滤流式工具调用中的尾部定界符，防止不完整的 JSON 片段污染
def filter_delta_text(
    delta_text: str,
    previous_text: str,
) -> tuple[str, bool]:
    bracket_level = _bracket_level(previous_text)
    updated_delta = ""
    passed_zero = False
    for char in delta_text:
        if char == "{":
            bracket_level += 1
            passed_zero = bracket_level == 0
        elif char == "}":
            bracket_level -= 1
            passed_zero = bracket_level == 0
        # 只要层级不为 0 就保留字符
        if bracket_level != 0:
            updated_delta += char
        else:
            # 回到层级 0 且遇到逗号时截断
            if char == ",":
                break
    return updated_delta, passed_zero
```

vllm/entrypoints/openai/responses/streaming_events.py

新增 SimpleStreamingState 状态机和事件发射函数，是 Responses API 流式工具调用的事件生成核心

```
# vllm/entrypoints/openai/responses/streaming_events.py
# 状态枚举
class _StateType(Enum):
    NONE = auto()
```

```
    CONTENT = auto()
    REASONING = auto()
    TOOL_CALL = auto()

# 流式线程状态
@dataclass
class SimpleStreamingState:
    output_index: int = 0
    current_item_id: str = ""
    content_index: int = 0
    accumulated_text: str = ""
    tool_call_id: str = ""
    tool_call_name: str = ""
    tool_call_index: int | None = None
    has_emitted_tool_call_delta: bool = False
    current_state: _StateType = field(default_factory=lambda: _StateType.NONE)

# 开始输出消息内容
def emit_simple_content_open(
    state: SimpleStreamingState,
) -> list[StreamingResponsesResponse]:
    state.current_state = _StateType.CONTENT
    state.current_item_id = random_uuid()
    state.content_index = 0
    state.accumulated_text = ""
    return [
        ResponseOutputItemAddedEvent(
            type="response.output_item.added",
            sequence_number=-1,
            output_index=state.output_index,
            item=ResponseOutputMessage(
                id=state.current_item_id,
                type="message",
                role="assistant",
                content=[],
                status="in_progress",
            ),
        ),
        ResponseContentPartAddedEvent(
            type="response.content_part.added",
            sequence_number=-1,
            output_index=state.output_index,
            item_id=state.current_item_id,
            content_index=state.content_index,
            part=ResponseOutputText(
                type="output_text",
                text="",
                annotations=[],
                logprobs=[],
```

```
            ),
        ),
    ]
```

评论区精华

Review 过程中，主要讨论集中在以下几点：

- history_tool_call_cnt 未递增：gemini-code-assist[bot] 和 sfeng33 指出，在 Parser.parse_delta 中 state.history_tool_call_cnt 被传递给 _extract_tool_calls_streaming 但从未递增，可能导致多工具调用时 ID 冲突。作者回复确认尚未处理此逻辑，将在后续 PR 中修复。
- 硬编码模型路径：gemini-code-assist[bot] 发现测试 conftest.py 中使用了本地路径 /mnt/data4/models/Qwen/Qwen3-8B，建议使用公开模型名称或可配置 fixture。此问题未收到回复。
- 生产代码中的 debug print：responses/serving.py 中残留了 print(f"Delta message:---------- {delta_message}")，被指出应删除。
- 未初始化 ID 问题：流式响应处理器中 current_item_id 被初始化为空字符串但未及时更新，可能导致第一个输出项的 ID 为空；且工具调用 ID 使用存在不一致，先计算 current_tool_call_id 却使用了 tool_call_id。

最终 sfeng33 批准了 PR，部分问题虽被指出但未在本次修复，作者计划后续处理，反映了快速迭代的风格。

- history_tool_call_cnt 未递增 (correctness): 作者承认尚未处理，将在后续 PR 中修复。PR 仍被批准合并。
- 硬编码模型路径 (testing): 未收到回复，测试仍包含硬编码路径。
- Debug print 残留 (other): 应在移除打印后合并。
- 工具调用 ID 使用不一致 (correctness): 评论未收到回复，代码可能仍存在问题。

风险与影响

- 风险：
  1. 回归风险：responses/serving.py 中删除了 573 行原处理逻辑并替换为新的状态机模型，虽然设计更清晰，但流式响应逻辑复杂，可能引入边缘情况下的行为差异或事件序列错误。
  2. 工具调用状态管理：abstract_parser.py 中新增的 history_tool_call_cnt 未递增的问题已在 Review 中指出，如果在新工具调用发生时计数器未更新，可能导致 tool_call_idx 重复，影响工具调用 ID 的唯一性（尽管 random_uuid 也可用，但索引可能用于本地调试）。
  3. 多模块耦合：新模块 streaming.py 同时被聊天补全和 Responses API 依赖，任何对该模块的修改可能同时影响两个 API，需要更谨慎的测试。
  4. 测试覆盖：仅新增了一个参数化测试用例，覆盖 required 和 auto 场景，但未覆盖 named 工具选择、Mistral tokenizer 路径、异常恢复等情况。硬编码模型路径也导致测

试无法在 CI 或其他环境运行。 - 影响：用户影响：使用 Responses API 且需要流式工具调用的用户现在可以正常工作，尤其是 tool_choice='required' 的请求会获得符合预期的 SSE 事件序列。对于聊天补全 API，内部重构对外部行为无影响。系统影响：代码量净减（+932/-718），模块化程度提高。新的 streaming.py 成为工具调用流式解析的中心，便于未来功能扩展。Responses API 流式处理架构从大规模内联代码转向状态机 + 事件发射模式，更易于推理和维护。团队影响：功能新增由社区贡献者完成，review 指出了几个值得后续修复的问题，为后续贡献者提供了改进方向。

- 风险标记：工具调用状态未递增，多模块耦合，测试覆盖不足，硬编码测试路径

关联脉络

- 暂无明显关联 PR