PR #25881 完整报告

sgl-project/sglang

Fix Responses API request handling

合并时间：2026-06-13 05:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25881

执行摘要

- 一句话：修复并增强 Responses API，新增函数工具与流式 SSE
- 推荐动作：强烈建议精读：本 PR 是 SGLang 中 Responses API 从"概念验证"走向"生产可用"的关键合并。它展示了如何将多个贡献者的修复高效整合、通过细致的代码评审（alexnails 的深度评论）识别潜在生产问题、以及如何设计兼容 OpenAI 规范的端点。serving_responses.py 中的输入规范化、流式生成和工具提取逻辑值得深入理解。

功能与动机

/v1/responses 端点在请求验证、多模态输入、会话历史回放、用法统计和工具支持方面存在多处严重问题，直接导致无效负载报 500、token 统计始终为零、多模态请求失败、函数工具调用返回原始文本等。本 PR 整合了 #25691、#20172、#24113 等多个孤立修复，旨在一次性解决全部已知缺陷，使 Responses API 符合 OpenAI 规范并可投入生产使用。

实现拆解

1. 请求验证与协议定义扩展— 在 protocol.py 中扩展 ResponseTool 类型以接受完整的 OpenAI Responses 工具种类，新增 validate_function_tool 校验器，添加入口规范化方法 normalize_responses_input，放宽 input 类型以接受宽松的字典格式。
2. 输入消息构建规范化— 在 serving_responses.py 中新增 _normalize_response_content_part_for_chat、_normalize_response_message_for_chat 等函数，将 Responses API 特有的 input_text、input_image 等内容部分转换为 Chat 模板格式；_merge_consecutive_assistant_messages 合并连续助手消息。
3. 前一次回应回放修复— 重写 _construct_input_messages 中 previous_response_id 的处理逻辑，正确提取 ResponseOutputMessage 的文本内容作为助手上文，跳过 reasoning、function_call 和 refusal 项。
4. 函数工具支持— 新增 _response_tools_to_chat_tools 将 Responses 工具转换为 Chat 工具；在生成输出项时通过 FunctionCallParser 从生成文本中解析原生函数调用，返回 ResponseFunctionToolCall 而非原始文本。
5. 流式 SSE 生成— 实现 responses_stream_generator_non_harmony，产生符合 OpenAI Responses 格式的类型化 SSE 事件（response.created、output_item.added、output_text.delta 等），并新增 _is_thinking_enabled_for_request、

_wants_reasoning_summary 控制推理摘要流。 配套改动：修正 responses_full_generator 中用法统计从 meta_info 字典提取（之前使用 hasattr 始终失败）；在 http_server.py 中将 /v1/responses 路由绑定到 ResponsesRequest；新增超过 1000 行的单元测试覆盖消息构造、流式事件顺序、协议验证和用法统计。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_responses.py（模块 响应服务；类别 source；类型 core-logic；符号 _wants_reasoning_summary, _is_thinking_enabled_for_request, _response_tools_to_chat_tools, _normalize_response_content_part_for_chat）：Responses API 请求处理的入口，包含建消息构造、流式生成、工具解析、用法统计等全部核心逻辑，修改量最大（+1117/-69）。
- python/sglang/srt/entrypoints/openai/protocol.py（模块 请求协议；类别 source；类型 data-contract；符号 validate_function_tool, normalize_responses_input, _normalize_input_item_for_validation, _normalize_content_part_for_validation）：定义 Responses API 的请求模型，扩展工具类型支持全 OpenAI 规范，新增输入规范化与函数工具验证器。
- test/registered/unit/entrypoints/openai/test_serving_responses.py（模块 响应测试；类别 test；类型 test-coverage；符号 InputMessageConstructionTestCase, test_previous_response_replays_assistant_text_not_instructions, test_input_parts_normalized_for_chat_templates, test_previous_response_id_input_list_does_not_call_copy_module）：新增针对消息构造、输入规范化、前后文回放、工具转发的全面单元测试，覆盖核心回归场景。
- test/registered/unit/entrypoints/openai/test_serving_responses_stream.py（模块 流式测试；类别 test；类型 test-coverage；符号 _StreamFixture, NonHarmonyStreamTestCase, test_emits_typed_sse_events_in_order, test_required_tool_choice_emits_function_call_events）：非 Harmony 模型流式 SSE 生成的功能测试，验证事件顺序、工具调用 SSE 事件和输出顺序。
- test/registered/unit/entrypoints/openai/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 MockTokenizerManager, MockTemplateManager, make_serving, collect_stream_events）：提供测试基础设施（mock 管理器、make_serving 工厂、流事件收集工具），被所有响应测试文件共享。

关键符号：_wants_reasoning_summary, _is_thinking_enabled_for_request, _response_tools_to_chat_tools, _normalize_response_content_part_for_chat, _normalize_response_message_for_chat, _collect, _output_message_text, _merge_consecutive_assistant_messages, validate_function_tool, normalize_responses_input, _normalize_input_item_for_validation, _normalize_content_part_for_validation, responses_stream_generator_non_harmony

关键源码片段

python/sglang/srt/entrypoints/openai/serving_responses.py

Responses API 请求处理的入口，包含建消息构造、流式生成、工具解析、用法统计等全部核心逻辑，修改量最大（+1117/-69）。

```
# 在 create_responses 开头检查 tool_choice 约束
# tool_choice="required" 仅能用于 type="function" 的工具
# 其他内置工具类型（web_search、code_interpreter 等）无法被强制选择
if request.tool_choice == "required" and not any(
    tool.type == "function" for tool in (request.tools or [])
):
    return self.create_error_response(
        'tool_choice="required" requires at least one tool with '
        'type="function"; other built-in tool types cannot be forced.'
    )

# 构建 GenerateReqInput 时，从 processed_messages 中提取多模态数据
# 使用 truthiness 判断，替代之前的条件表达式
image_data=processed_messages.image_data if processed_messages else None,
video_data=processed_messages.video_data if processed_messages else None,
audiodata=processed_messages.audio_data if processed_messages else None,
modalities=processed_messages.modalities if processed_messages else None,
```

python/sglang/srt/entrypoints/openai/protocol.py

定义 Responses API 的请求模型，扩展工具类型支持全 OpenAI 规范，新增输入规范化与函数工具验证器。

```
class ResponseTool(BaseModel):
    # 完整的 OpenAI Responses 工具类型集合
    # 仅 function / web_search* / code_interpreter 有实际执行路径
    # 其余类型仅通过校验，不拒绝客户端
    type: RESPONSE_TOOL_TYPES = Field(description="Type of tool to enable")
    name: Optional[str] = None
    description: Optional[str] = None
    parameters: Optional[Dict[str, Any]] = None
    strict: bool = False
    # namespace 工具内部可以包含子工具列表
    tools: Optional[List[Dict[str, Any]]] = None

    @model_validator(mode="after")
    def validate_function_tool(self) -> "ResponseTool":
        # function 类型必须提供 name，否则拒绝
        if self.type == "function" and not self.name:
            raise ValueError("Function tools must include a name.")
        return self
```

test/registered/unit/entrypoints/openai/test_serving_responses_stream.py

非 Harmony 模型流式 SSE 生成的功能测试，验证事件顺序、工具调用 SSE 事件和输出顺序。

```
def test_emits_typed_sse_events_in_order(self):
    serving = make_serving()
    serving.reasoning_parser = None
```

```
        serving.tool_call_parser = None
        request = ResponsesRequest(model="x", input="hi", stream=True, store=False)
        fixture = _StreamFixture(serving, request)
        events = fixture.run(
            [
                _engine_chunk("Hel", 1),
                _engine_chunk("Hello", 2),
                _engine_chunk("Hello world", 4, finish=True),
            ]
        )
        types = event_types(events)
        # 第一个事件必须是 response.created，最后一个是 response.completed
        self.assertEqual(types[0], "response.created")
        self.assertEqual(types[-1], "response.completed")
        # 中间必须包含 output_item.added、content_part.added、output_text.delta 等
        for ev in (
            "response.output_item.added",
            "response.content_part.added",
            "response.output_text.delta",
            "response.output_text.done",
            "response.content_part.done",
            "response.output_item.done",
        ):
            self.assertIn(ev, types)
        # 验证 sequence_number 连续递增
        seqs = [p["sequence_number"] for p in event_payloads(events)]
        self.assertEqual(seqs, list(range(len(seqs))))
```

评论区精华

1. 内存无限制增长风险— alexnails 指出 response_store 和 msg_store 是未加限制的进程内字典，store 默认 True，长期使用将耗尽内存。该问题属于 pre-existing，但 PR 使端点真正可用而扩大了暴露面。决定在后续 issue 中跟进 LRU/TTL 限制（status: acknowledged）。
2. 同步 tokenizer.encode 阻塞事件循环— alexnails 发现 create_responses 中通过 len(tokenizer.encode(...)) 计算 default_max_tokens 是同步 HF 操作，会阻塞 asyncio 循环。PR 已改进部分路径避免重复 tokenize，但核心的 encode 调用仍在循环内（status: partially addressed）。
3. 工具类型扩展— Kontinuation 建议将 web_search 工具类型一并加入避免后续反复。JustinTong0323 在后续提交中将 ResponseTool.type 扩展到完整 OpenAI 工具集，包括 namespace、mcp、file_search 等（status: resolved）。
4. 代码 DRY 简化— gemini-code-assist[bot] 建议对 processed_messages 条件赋值使用真值检查，并将 meta_info 提取合并为单一代码块。JustinTong0323 在 commit 882b1f7 中采纳（status: resolved）。
5. Qwen3 推理检测潜在误判— alexnails 担忧在 Qwen3 模型中若 <tool_call> 出现在推理内容中，更新后的推理检测器会强制关闭 think，可能导致普通文本被解析为工具调用。

JustinTong0323 解释了该行为的合理性及限制（status: resolved）。

- response_store 无限制增长导致内存泄漏 (performance): 该问题属于 pre-existing，但 PR 使端点可用从而扩大暴露面。作者确认需要后续添加 LRU/TTL 限制，当前建议用户在请求中显式设置 store: false。
- 同步 tokenizer.encode 阻塞 asyncio 事件循环 (performance): 已部分改进（使用已编码列表长度避免重复 tokenize 等），但核心编码调用仍在循环内。作者表示可后续进一步优化。
- 工具类型扩展至完整 OpenAI 规范 (design): JustinTong0323 扩展 ResponseTool.type 至 OpenAI 完整工具集（function, web_search, code_interpreter, file_search, namespace, mcp 等），并添加 validate_function_tool 校验器。
- Qwen3 推理检测器隐式关闭 think 的潜在误判 (correctness): JustinTong0323 解释了 Qwen3 模板规范中 <tool_call> 意味着推理结束，此行为符合预期；限制为该系列模型，不会影响其他架构。
- gemini-code-assist 建议的条件赋值简化 (style): JustinTong0323 在 commit 882b1f7 中采纳建议，并推广至四个数据字段和 stop/tool_constraint 参数。

风险与影响

- 风险:
  1. 内存泄漏风险— serving_responses.py 中的 response_store 和 msg_store 为未限定的进程内字典，无 TTL/ 淘汰策略；store 默认为 True，每个请求都会无限制累积数据。生产部署需注意，建议设置 store: false 或跟随后续修复。
  2. 性能风险— serving_responses.py 在 create_responses 的请求循环中同步调用 tokenizer.encode 计算 default_max_tokens，会阻塞 asyncio 事件循环，长上下文多模态请求可能导致延迟抖动。
  3. 类型兼容性风险— protocol.py 中 input 字段类型放宽为 List[Dict[str, Any]]，可能掩盖无效负载的类型错误，依赖下游规范化处理。
  4. 推理检测误判风险— reasoning_parser.py 中 Qwen3 检测器将 <tool_call> 视为推理关闭标记，若模型在推理流中产生该字面串，可能错误截断推理并将内容送入工具解析器。
  5. 测试覆盖缺口— 单元测试主要集中在非 Harmony 路径；Harmony 路径的流式 SSE 和工具调用提取缺乏直接覆盖。 - 影响：用户影响：使用 /v1/responses 的开发者将获得正确的用法统计、函数工具输出、多模态输入处理、会话历史回放以及兼容 OpenAI 规范的流式 SSE。修复前无法使用的客户端（如 Codex CLI）现在可正常工作。系统影响：无破坏性变更，但新的请求存储机制可能增加内存压力，需在生产监控中注意。团队影响：降低了 Responses API 的维护负担，将多个零散修复集成至统一代码库；后续需跟进内存管理和性能优化。
- 风险标记：内存泄漏风险，同步 tokenize 性能瓶颈，推理检测潜在误判，Harmony 路径测试覆盖不足

关联脉络

- PR #25691 [Qwen3.6] Fix /v1/responses for multimodal models: 被完全整合：修复多模态模型返回 input_ids 而非 text 导致 normalize 失败的问题。

- PR #20172 fix: correct dict key check for usage accounting in /v1/responses: 被完全整合：修正 usage 统计从 hasattr 改为字典键检查，确保 token 计数正确。
- PR #24113 [Bugfix] Replay prior assistant output for previous_response_id: 被完全整合：重放前一次回应时正确提取 assistant 文本而非 instructions。
- PR #23766 Codex CLI Responses interop: 本 PR 的演进覆盖了该 PR 的改动，合并后原作者已验证 Codex CLI 可以正常工作。