PR #45396 完整报告

vllm-project/vllm

[Frontend] Support strict mode for tool calling with ResponsesAPI

合并时间：2026-06-12 23:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45396

执行摘要

- 一句话：为 ResponsesAPI 扩展严格工具调用模式
- 推荐动作：该 PR 值得精读，尤其是 structural_tag_registry.py 中的类型转换函数，展示了如何为不同 API 协议适配 xgrammar。建议后续增加端到端的严格模式测试，并关注 prepare_structured_tag 的兼容性影响。

功能与动机

跟进 PR #45003，为 ResponsesAPI 提供相同的严格工具调用模式，确保工具调用时结构化输出的正确性。

实现拆解

1. 扩展类型定义和转换函数：在 vllm/tool_parsers/structural_tag_registry.py 中添加对 OpenAI Responses 类型的导入（FunctionTool、ResponsesToolChoice 等），并新增 _dump_tool_for_xgrammar、_dump_tool_choice_for_xgrammar、_dump_allowed_tool_ref_for_xgrammar 函数，将 Responses 对象转换为 xgrammar 所需的字典协议。
2. 修改核心函数 get_model_structural_tag：将其 tools 参数类型从 list[ChatCompletionToolsParam] | None 扩展为 Sequence[ChatCompletionToolsParam | ResponsesTool] | None，并内部调用新的转储函数。
3. 调整 _apply_structural_tag：在 vllm/parser/abstract_parser.py 中，移除对 ChatCompletionRequest 的类型检查，使该函数同时适用于 ResponsesRequest；对于 ResponsesRequest，将 request.text 设置为 None 而非 request.response_format = None，以确保结构化输出正确生效。
4. 更新 get_structural_tag 签名：在 vllm/tool_parsers/abstract_tool_parser.py 中，使 get_structural_tag 方法接受 ChatCompletionRequest | ResponsesRequest 联合类型。
5. 测试配置调整：在 tests/entrypoints/openai/responses/conftest.py 中，移除 VLLM_ENFORCE_STRICT_TOOL_CALLING=0 环境变量，使严格模式在测试中默认启用（除非被覆盖）。 另外，vllm/reasoning/abs_reasoning_parsers.py 中 prepare_structured_tag 的返回值从 None 改为 original_tag，这是一个兼容性修复，确保结构标记在推理解析器中被正确传递。

关键文件：

- vllm/tool_parsers/structural_tag_registry.py（模块 工具解析器；类别 source；类型 core-logic；符号 _dump_tool_for_xgrammar, _dump_tool_choice_for_xgrammar, _dump_allowed_tool_ref_for_xgrammar, ToolChoice）：核心逻辑文件；添加了对 OpenAI Responses 类型的支持，并重构了模型转储函数。
- vllm/parser/abstract_parser.py（模块 解析器；类别 source；类型 core-logic；符号 _apply_structural_tag）：调整了 _apply_structural_tag 以支持 ResponsesRequest，并修改了结构化输出的清理逻辑。
- vllm/tool_parsers/abstract_tool_parser.py（模块 工具解析器；类别 source；类型 core-logic；符号 get_structural_tag）：更新了 get_structural_tag 方法签名以接受 ResponsesRequest，使子类能够正确处理两种请求类型。
- vllm/reasoning/abs_reasoning_parsers.py（模块 推理解析；类别 source；类型 core-logic；符号 prepare_structured_tag）：修改 prepare_structured_tag 默认行为，从返回 None 改为返回 original_tag，确保结构标记在推理链中被传递。
- tests/entrypoints/openai/responses/conftest.py（模块 测试配置；类别 test；类型 test-coverage）：移除了显式禁用严格工具调用的环境变量，使测试默认启用严格模式。

关键符号：_dump_tool_for_xgrammar, _dump_tool_choice_for_xgrammar, get_model_structural_tag, _apply_structural_tag, get_structural_tag, prepare_structured_tag

关键源码片段

vllm/tool_parsers/structural_tag_registry.py

核心逻辑文件；添加了对 OpenAI Responses 类型的支持，并重构了模型转储函数。

```
def _dump_tool_for_xgrammar(
    tool: ChatCompletionToolsParam | ResponsesTool,
) -> dict[str, Any]:
    """
    将 vLLM 的 tool 对象转换为 xgrammar 可接受的字典格式。
    对于 ResponsesAPI 的 FunctionTool，直接构造字典；
    对于 ChatCompletionToolsParam，则调用 model_dump。
    """
    if isinstance(tool, FunctionTool):
        # ResponsesAPI 的 FunctionTool 使用直接属性
        function: dict[str, Any] = {"name": tool.name}
        if tool.description is not None:
            function["description"] = tool.description
        if tool.parameters is not None:
            function["parameters"] = tool.parameters
        if tool.strict is not None:
            function["strict"] = tool.strict
        return {"type": "function", "function": function}
    # Chat Completions API 的类型使用 Pydantic model_dump
    dumped_tool = tool.model_dump(mode="json", exclude_none=True)
    if isinstance(tool, ChatCompletionToolsParam):
```

```
            # 兼容 xgrammar 的 FunctionToolParam 格式
            return {
                "type": "function",
                "function": {
                    "name": dumped_tool.get("function", {}).get("name", ""),
                    "parameters": dumped_tool.get("function", {}).get("parameters", {}),
                },
            }
        return dumped_tool
```

vllm/parser/abstract_parser.py

调整了 _apply_structural_tag 以支持 ResponsesRequest，并修改了结构化输出的清理逻辑。

```
    def _apply_structural_tag(
        self, request: ChatCompletionRequest | ResponsesRequest
    ) -> ChatCompletionRequest | ResponsesRequest:
        # 不再要求必须是 ChatCompletionRequest
        if (
            self._tool_parser is None
            or self._tool_parser.structural_tag_model is None
            or not request.tools
        ):
            return request

        need_tool_calling = (
            request.tool_choice == "auto"
            or request.tool_choice == "required"
            or isinstance(
                request.tool_choice,
                (ChatCompletionNamedToolChoiceParam, ToolChoiceFunction), # 新增
                ToolChoiceFunction
            )
        )
        if not need_tool_calling:
            return request

        structure_tag = self._tool_parser.get_structural_tag(
            request,
            reasoning=False,
        )
        if structure_tag is None:
            return request

        structural_tag = json.dumps(structure_tag.model_dump())
        request.structured_outputs = StructuredOutputsParams(
            structural_tag=structural_tag,
        )
        # 对 ResponsesRequest 清空 text，对 ChatCompletionRequest 清空 response_format
        if isinstance(request, ResponsesRequest):
```

```
            request.text = None
        else:
            request.response_format = None
        return request
```

评论区精华

仅有一个批准评论（sfeng33: “LGTM, thanks for the work!”），无实质性讨论或争议。

- PR 审批 (other): 无变化

风险与影响

- 风险：
  1. API 兼容性风险：ResponsesAPI 严格模式的引入可能改变已有依赖旧行为的用户的工具调用输出格式，要求 xgrammar 支持相应模型。
  2. 测试覆盖不足：仅移除了 conftest 中的环境变量，缺乏端到端的严格模式测试用例，可能遗漏边界情况。
  3. 解析器行为变化：abs_reasoning_parsers.py 中 prepare_structured_tag 的行为从返回 None 变为返回 original_tag，可能影响其他推理解析器的逻辑，尽管目前此类解析器不多。 - 影响：用户影响：使用 ResponsesAPI 进行工具调用的用户将自动获得严格模式的结构化输出保证（通过 VLLM_ENFORCE_STRICT_TOOL_CALLING 环境变量控制）。系统影响：无显著性能开销；增加了对 OpenAI Responses 类型的依赖。团队影响：需要为两个 API 家族（Chat Completions 和 Responses）维护平行的严格模式配置。 - 风险标记：API 兼容性风险，测试覆盖不足，解析器行为变化

关联脉络

- PR #45003 [Frontend] Support strict mode for tool calling: 本 PR 扩展了 #45003 为 Chat Completions API 实现的严格工具调用模式至 ResponsesAPI。