PR #45147 完整报告

vllm-project/vllm

[Bugfix] Fix tool parsing crash with non-function tool types (e.g. WebSearchTool)

合并时间：2026-06-11 01:17

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45147

执行摘要

- 一句话：修复非函数工具导致解析崩溃
- 推荐动作：值得快速合入，修复了明确的崩溃 bug，测试覆盖充分。

功能与动机

Codex CLI 默认工具定义包含非函数类型（如 web_search），当模型生成工具调用时，多个工具解析器因 TypeError 崩溃。需要过滤非函数工具以确保兼容性。

实现拆解

1. 新增过滤辅助函数：在 vllm/tool_parsers/utils.py 新增 _is_function_tool(tool) 函数，判断工具是否属于 FunctionTool 或 ChatCompletionToolsParam 类型。
2. 修改 find_tool_properties：遍历工具列表时，先过滤非函数工具，跳过 _is_function_tool 返回 False 的工具，避免向 _extract_tool_info 传递不支持的类型。
3. 修改 _get_json_schema_from_tools：构建 anyOf 列表前，先用 _is_function_tool 过滤 fn_tools，只对函数工具生成 schema，同时向 _get_tool_schema_defs 传入 fn_tools，避免处理非函数工具的 $defs。
4. 新增测试类 TestNonFunctionToolsSkipped：在 tests/tool_use/test_tool_choice_required.py 添加 3 个测试方法，覆盖纯非函数工具、混合工具场景，验证 find_tool_properties 和 get_json_schema_from_tools 正确跳过非函数工具。

关键文件：

- vllm/tool_parsers/utils.py（模块 工具解析；类别 source；类型 core-logic；符号 _is_function_tool）：内核变更文件，新增 _is_function_tool 过滤函数，修改 find_tool_properties 和 _get_json_schema_from_tools 以跳过非函数工具。
- tests/tool_use/test_tool_choice_required.py（模块 工具测试；类别 test；类型 test-coverage；符号 TestNonFunctionToolsSkipped, test_find_tool_properties_skips_web_search, test_find_tool_properties_only_non_function_tools, test_get_json_schema_with_mixed_tools）：新增测试覆盖，验证非函数工具被正确跳过，确保回归安全。

关键符号：_is_function_tool, find_tool_properties, _get_json_schema_from_tools

关键源码片段

vllm/tool_parsers/utils.py

内核变更文件，新增 _is_function_tool 过滤函数，修改 find_tool_properties 和 _get_json_schema_from_tools 以跳过非函数工具。

```
# vllm/tool_parsers/utils.py

# 新增过滤函数，判断工具是否为函数工具
def _is_function_tool(tool: Tool) -> bool:
    # FunctionTool 对应 Responses API，ChatCompletionToolsParam 对应 Chat API
    return isinstance(tool, (FunctionTool, ChatCompletionToolsParam))


def find_tool_properties(
    tools: list[Tool] | None,
    tool_name: str,
) -> dict[str, Any]:
    """Find a tool by name and return its properties dict, or {}."""
    if not tools:
        return {}
    for tool in tools:
        if not _is_function_tool(tool):
            continue # 跳过非函数工具，避免 TypeError
        name, params = _extract_tool_info(tool)
        if name == tool_name:
            return (params or {}).get("properties", {})
    return {}


def _get_json_schema_from_tools(
    tools: list[Tool],
) -> dict:
    # 只筛选函数工具，非函数工具不参与 schema 生成
    fn_tools = [t for t in tools if _is_function_tool(t)]
    json_schema = {
        "type": "array",
        "minItems": 1,
        "items": {
            "type": "object",
            "anyOf": [_get_tool_schema_from_tool(tool) for tool in fn_tools],
        },
    }
    json_schema_defs = _get_tool_schema_defs(fn_tools)
    if json_schema_defs:
        json_schema["$defs"] = json_schema_defs
    return json_schema
```

tests/tool_use/test_tool_choice_required.py

新增测试覆盖，验证非函数工具被正确跳过，确保回归安全。

```
# tests/tool_use/test_tool_choice_required.py

from openai.types.responses import FunctionTool, WebSearchTool

FUNCTION_TOOL = FunctionTool(
    type="function",
    name="get_weather",
    parameters={
        "type": "object",
        "properties": {"city": {"type": "string"}},
        "required": ["city"],
    },
)
WEB_SEARCH_TOOL = WebSearchTool(type="web_search")


class TestNonFunctionToolsSkipped:
    """Non-function tools (web_search, etc.) must be silently skipped
    by the tool-schema utilities instead of raising TypeError."""

    def test_find_tool_properties_skips_web_search(self):
        # 混合工具：应只匹配函数工具
        tools = [WEB_SEARCH_TOOL, FUNCTION_TOOL]
        props = find_tool_properties(tools, "get_weather")
        assert props == {"city": {"type": "string"}}

    def test_find_tool_properties_only_non_function_tools(self):
        # 只有非函数工具时返回空字典
        props = find_tool_properties([WEB_SEARCH_TOOL], "get_weather")
        assert props == {}

    def test_get_json_schema_with_mixed_tools(self):
        # 混合工具生成 schema 时只包含函数工具
        tools = [WEB_SEARCH_TOOL, FUNCTION_TOOL]
        schema = get_json_schema_from_tools(tools=tools, tool_choice="required")
        assert isinstance(schema, dict)
        any_of = schema["items"]["anyOf"]
        assert len(any_of) == 1
        assert any_of[0]["properties"]["name"]["enum"] == ["get_weather"]
```

评论区精华

无 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：改变小，过滤逻辑只作用于非函数工具，对正常函数工具路径无影响。风险很低。
- 影响：修复了使用非函数工具（如 web_search）时解析器崩溃的问题，影响使用 Codex CLI 或类似混合工具定义的用户。对仅使用函数工具的用户无影响。
- 风险标记：低风险

关联脉络

- 暂无明显关联 PR