PR #50334 完整报告

vllm-project/vllm

[Bugfix][Responses] Add tests for Chat Completions Responses API Render Parity

合并时间：2026-08-01 03:45

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50334

执行摘要

- 一句话：新增双 API 渲染一致性测试并修复工具渲染差异
- 推荐动作：值得精读，重点是 test_render_parity.py 的捕获 - 对比测试设计：它没有 mock 掉整个预处理链路，只在 HF 渲染边界安装桩，保留了两条 API 各自预处理逻辑的真实性，这种 " 边界打桩 " 思路对其他跨 API 一致性测试有借鉴价值。也值得关注两个序列化器的白名单过滤与 construct_tool_dicts 的参数化设计。对后续要扩展 Responses API 能力的开发者，这套测试应作为默认配套。

功能与动机

PR body 声明目标是为 Chat Completions 与 Responses 之间的渲染一致性添加测试，覆盖多轮对话、tools（strict、defer_loading、extra 字段）、tool_choice、reasoning_effort/enable_thinking 以及 template kwargs。渲染入口的分叉会直接影响提示词模板输出，进而影响工具调用正确性与响应质量。测试驱动下发现并修复两个不一致：一是 Responses 的 FunctionTool 与 ChatCompletionToolsParam 渲染结果不一致（该问题源于上游 PR#49824 的讨论），二是 Responses 的 tool_choice='none' 未像 Chat Completions 那样尊重 --exclude-tools-when-tool-choice-none。行为变化随之产生：工具中的 extra 自定义字段被忽略、tool_choice='none' 时默认把工具注入 prompt。

实现拆解

变更入口是新增的 tests/entrypoints/openai/test_render_parity.py：它通过 RenderCapture 在 render_chat_async 这一 HF 渲染边界安装桩函数，记录两套 API 各自预处理后产出的 messages 与 ChatParams，再由 _assert_parity 以配对请求做逐字段断言。关键设计是没有 mock 掉 OnlineRenderer.preprocess_chat 链路，只替换最末端的渲染函数，因此 Chat Completions 与 Responses 各自的工具转换、消息构造逻辑都真实执行。

实现过程按以下步骤展开：

1. 建立跨 API 渲染捕获基础设施：新增 test_render_parity.py（479 行），MockModelConfig/MockHFConfig 提供构造 OnlineRenderer 与 OpenAIServingResponses 所需的完整配置；_capture_chat 走 online_renderer.render_chat，_capture_responses 走 serving._make_request，两者最终汇聚到同一个桩函数，保证观测点在渲染边界完全一致。

2. 修复 FunctionTool 渲染不一致：vllm/entrypoints/openai/responses/utils.py 的 convert_tool_responses_to_completions_format 从返回裸 dict 改为构造 ChatCompletionToolsParam(type="function", function=FunctionDefinition.model_validate(...))，使 Responses 工具 schema 复用 Chat Completions 的字段校验与序列化；construct_tool_dicts 对应改为对转换结果调用 .model_dump()。配套在 vllm/entrypoints/openai/engine/protocol.py 的 FunctionDefinition._serialize 与 vllm/entrypoints/openai/chat_completion/protocol.py 的 ChatCompletionToolsParam._serialize 中加入按 model_fields 白名单过滤的逻辑，strict、defer_loading 为 None 时仍按原逻辑剔除——这是 extra 字段被统一忽略的实现来源。
3. 对齐 tool_choice='none' 语义：vllm/entrypoints/openai/responses/serving.py::_make_request 调用 construct_tool_dicts 时新增 exclude_tools_when_tool_choice_none=self.online_renderer.exclude_tools_when_tool_choice_none 透传，construct_tool_dicts 的判定条件由 tool_choice == "none" 改为 tool_choice == "none" and exclude_tools_when_tool_choice_none，默认行为随之变为把工具写进 prompt。
4. 收敛非 function 工具并标记缺口：vllm/tool_parsers/utils.py::iter_response_function_tool_dicts 由无条件的 else 分支收紧为 elif isinstance(tool, FunctionTool)，type="code_interpreter" 等非 function 工具不再被注入；该收紧暴露了 MCP 工具在 ParsableContext 路径从未被正确渲染的既有缺口，最终对 tests/entrypoints/openai/responses/test_parsable_context.py::test_mcp_tool_call 加 xfail 并注明原因。
5. 测试配套收尾：tests/entrypoints/openai/responses/test_responses_utils.py 删除旧的裸 dict 断言用例（test_convert_tool_responses_to_completions_format），覆盖职责移交给新的 parity 测试；test_render_parity.py 共 15 个用例，覆盖多轮对话、tools（strict、defer_loading、extra）、tool_choice（auto/none/required）、reasoning_effort、enable_thinking 与 template kwargs。

关键文件：

- tests/entrypoints/openai/test_render_parity.py（模块 渲染测试；类别 test；类型 test-coverage；符号 MockHFConfig, MockModelConfig, get_diff_sampling_param, CapturedRenderInputs）：新增 479 行跨 API 渲染一致性测试，是本次变更的核心资产：RenderCapture 在 render_chat_async 边界捕获输入，_assert_parity 对两套 API 的 messages 与 ChatParams 逐字段断言，驱动了全部 bugfix。
- vllm/entrypoints/openai/responses/utils.py（模块 响应处理；类别 source；类型 core-logic；符号 convert_tool_responses_to_completions_format, construct_tool_dicts）：核心修复所在：convert_tool_responses_to_completions_format 改为构造 ChatCompletionToolsParam，construct_tool_dicts 增加 exclude_tools_when_tool_choice_none 参数，直接决定工具 schema 如何进入渲染管线。
- tests/entrypoints/openai/responses/test_responses_utils.py（模块 响应测试；类别 test；类型 test-coverage；符号 test_convert_tool_responses_to_completions_format）：删除旧的裸 dict 断言用例，说明转换逻辑的覆盖职责转移到了新的 parity 测试。
- vllm/entrypoints/openai/responses/serving.py（模块 响应服务；类别 source；类型 core-logic；符号 _make_request）：_make_request 将

exclude_tools_when_tool_choice_none 从 OnlineRenderer 透传给 construct_tool_dicts，是 tool_choice='none' 语义对齐的调用侧关键改动。

- vllm/entrypoints/openai/engine/protocol.py（模块 协议模型；类别 source；类型 core-logic；符号 FunctionDefinition._serialize）：FunctionDefinition._serialize 增加 model_fields 白名单过滤，是 extra 字段被统一忽略的实现来源之一。
- vllm/entrypoints/openai/chat_completion/protocol.py（模块 对话协议；类别 source；类型 core-logic；符号 ChatCompletionToolsParam._serialize）：ChatCompletionToolsParam._serialize 同步增加白名单过滤，保证与 FunctionDefinition 行为一致，是 extra 字段行为对称性的另一半。
- vllm/tool_parsers/utils.py（模块 工具解析；类别 source；类型 core-logic；符号 iter_response_function_tool_dicts）：iter_response_function_tool_dicts 收紧为仅处理 FunctionTool，修复类型泄漏但暴露 MCP 工具渲染缺口，是 review 讨论的焦点。
- tests/entrypoints/openai/responses/test_parsable_context.py（模块 上下文测试；类别 test；类型 test-coverage；符号 test_mcp_tool_call）：为 MCP 工具调用测试增加 xfail 标记，记录 ParsableContext 路径尚未支持 MCP 工具渲染的已知缺口。

关键符号：convert_tool_responses_to_completions_format, construct_tool_dicts, iter_response_function_tool_dicts, FunctionDefinition._serialize, ChatCompletionToolsParam._serialize, _make_request, _assert_parity, RenderCapture.take, test_multiturn_tool_calling

关键源码片段

tests/entrypoints/openai/test_render_parity.py

新增 479 行跨 API 渲染一致性测试，是本次变更的核心资产：RenderCapture 在 render_chat_async 边界捕获输入，_assert_parity 对两套 API 的 messages 与 ChatParams 逐字段断言，驱动了全部 bugfix。

```
class RenderCapture:
    """在 render_chat_async（HF 渲染边界）安装桩，记录其收到的输入。"""

    def __init__(self, online_renderer: OnlineRenderer) -> None:
        self.online_renderer = online_renderer
        self.captured: CapturedRenderInputs | None = None

        async def fake_render_chat_async(
            conversations, chat_params, tok_params=None, *,
            prompt_extras=None, skip_mm_cache=False,
        ):
            # 两条 API 路径最终都应只产生一组对话；
            # 记录消息与 ChatParams，再返回一个假 token 输入即可走完流程。
            assert len(conversations) == 1
            self.captured = CapturedRenderInputs(
                messages=list(conversations[0]),
                chat_params=chat_params,
            )
```

```
            return [list(conversations[0])], [tokens_input(prompt_token_ids=[0])]

        online_renderer.renderer.render_chat_async = fake_render_chat_async

    def take(self) -> CapturedRenderInputs:
        # 取走当前捕获并复位，保证每次调用只消费一次观测结果。
        assert self.captured is not None
        captured = self.captured
        self.captured = None
        return captured


async def _assert_parity(
    online_renderer: OnlineRenderer,
    serving: OpenAIServingResponses,
    *,
    chat_kwargs: dict[str, Any],
    responses_kwargs: dict[str, Any],
) -> None:
    """构造配对请求，分别捕获两个 API 的 HF 渲染输入，再断言完全相等。"""
    chat_req = ChatCompletionRequest(model=_MODEL, **chat_kwargs)
    responses_req = ResponsesRequest(model=_MODEL, **responses_kwargs)
    chat = await _capture_chat(online_renderer, chat_req)
    responses = await _capture_responses(serving, responses_req)

    # 消息本身必须一致，工具转换的差异会在这里直接暴露。
    assert chat.messages == responses.messages

    # 模板相关字段逐一对比：template、content format、媒体参数、
    # template kwargs（含 tools），全部相等才算渲染一致。
    chat_params = chat.chat_params
    responses_params = responses.chat_params
    assert chat_params.chat_template == responses_params.chat_template
    assert (
        chat_params.chat_template_content_format
        == responses_params.chat_template_content_format
    )
    assert chat_params.media_io_kwargs == responses_params.media_io_kwargs
    assert chat_params.mm_processor_kwargs == responses_params.mm_processor_kwargs
    assert dict(chat_params.chat_template_kwargs) == dict(
        responses_params.chat_template_kwargs
    )
```

vllm/entrypoints/openai/responses/utils.py

核心修复所在：convert_tool_responses_to_completions_format 改为构造 ChatCompletionToolsParam，construct_tool_dicts 增加 exclude_tools_when_tool_choice_none 参数，直接决定工具 schema 如何进入渲染管线。

```
def convert_tool_responses_to_completions_format(
```

```
    tool: dict,
) -> ChatCompletionToolsParam:
    """
    将 Responses API 的扁平工具 schema:
        {"type": "function", "name": "...", "description": "...", "parameters": {...}}
    转换为 Chat Completions 工具参数，供 chat-template 渲染使用。

    关键点：直接构造 ChatCompletionToolsParam 而不是返回裸 dict，
    从而复用 Chat Completions 侧的字段校验与序列化逻辑，
    保证两种 API 渲染时工具 schema 完全一致。
    """
    return ChatCompletionToolsParam(
        type="function",
        # 丢弃 `type` 键，其余字段交给 FunctionDefinition 严格校验，
        # 未知的 extra 字段会被 pydantic 白名单序列化过滤，
        # 与 Chat Completions 的渲染行为保持一致。
        function=FunctionDefinition.model_validate(
            {k: v for k, v in tool.items() if k != "type"}
        ),
    )


def construct_tool_dicts(
    tools: list[Tool],
    tool_choice: ToolChoice,
    exclude_tools_when_tool_choice_none: bool = False,
) -> list[dict[str, Any]] | None:
    # tool_choice="none" 时是否仍把工具注入 prompt 取决于服务端参数，
    # 与 Chat Completions 的 --exclude-tools-when-tool-choice-none 对齐。
    if not tools or (tool_choice == "none" and exclude_tools_when_tool_choice_none):
        return None
    return [
        convert_tool_responses_to_completions_format(tool).model_dump()
        for tool in iter_response_function_tool_dicts(tools)
    ]
```

评论区精华

核心讨论围绕 vllm/tool_parsers/utils.py 中 iter_response_function_tool_dicts 由 else 改为 elif isinstance(tool, FunctionTool) 引发的回归：code_interpreter 等 MCP 工具不再被注入，导致 tests/entrypoints/openai/responses/test_parsable_context.py::test_mcp_tool_call 断言失败（期望 mcp_call 出现在输出中，实际只有 reasoning 与 message）。yzong-rh 说明 MCP 工具在 ParsableContext 路径上从未被正确支持：旧路径只是注入一个无 name/parameters 的非法 function tool，新路径则完全不注入，属于 " 从坏的变成没有 "；他提出两个方案——xfail 该测试，或继续 dump 非 function 工具以保留旧行为。bbrowning 赞同 xfail 是正确做法，并指出 xfail 有助于未来在 ParsableContext 完善时清理。此外 bbrowning 在 approve 评论中认可测试套件对拦截这类回归的价值，并补充运行了 tool_parsers、responses 等一组周边单测确认无回归。

- 非 function 工具注入回归与 MCP 工具缺口 (correctness): 采纳 xfail 方案：对 test_mcp_tool_call 加 xfail 并在 reason 中注明 MCP 工具名与参数未被提取，留待 ParsableContext 完善后再清理。
- 渲染一致性测试作为回归护栏 (testing): 无条件合入，无未解决疑虑。

风险与影响

- 风险： 1) 行为变更风险：FunctionTool 与 ChatCompletionToolsParam 的 extra 字段现在会被 pydantic 静默丢弃（engine/protocol.py 与 chat_completion/protocol.py 的序列化器新增 model_fields 白名单过滤），依赖自定义扩展字段的工具方可能收到与之前不同的渲染结果，且是静默丢弃而非报错。 2) tool_choice='none' 语义变化：Responses API 默认将全部工具注入 prompt，prompt 变长会推高 token 消耗与 TTFT，依赖 --exclude-tools-when-tool-choice-none 才能恢复旧行为。 3) MCP 工具缺口只是被 xfail 掩蔽：ParsableContext 路径下 MCP 工具仍无法在 prompt 中渲染，真实场景会静默缺少 mcp_call 输出，存在回归风险。 4) 测试机制局限：RenderCapture 拦截的是 render_chat_async 边界，真实 HF 模板渲染、TokenizeParams、截断等不在覆盖范围，parity 测试全绿不代表端到端输出逐字节一致。
  - 影响：影响范围集中在 OpenAI 前端 API（Responses 与 Chat Completions 的工具渲染路径），不涉及推理内核与调度器。对用户：Responses API 的 tool_choice='none' 默认行为变化、extra 字段被忽略，属于可感知的行为变更；对团队：新增的 parity 测试为两个 API 的后续演进提供回归护栏，避免渲染再次分叉；对 API 实现者：FunctionDefinition 序列化现在按 model_fields 白名单输出，限制未知字段传导到模板，工具 schema 传递路径被显著收紧。
  - 风险标记：行为变更：extra 字段静默丢弃，行为变更：tool_choice=none 默认注入工具，MCP 工具缺口仅 xfail 掩蔽，测试覆盖渲染边界而非端到端输出

关联脉络

- PR #49824 FunctionTool not rendered identically to ChatCompletionToolsParam（PR #50334 body 引用）：PR body 明确说明本 PR 的第一个 bugfix（FunctionTool 渲染不一致）源于该 PR 的讨论，BFCL 端到端验证也挂在其上。
- PR #50515 [ROCm][CI] Restore Mistral tool-parser compatibility after unification: 同属 vllm/tool_parsers 路径的兼容性回归修复，说明工具渲染路径近期正在统一与收敛，本 PR 的 elif isinstance 收紧是该方向的一部分。
- PR #50491 [Bugfix][Frontend] Raise VLLMValidationError for user-facing errors in chat_utils.py: 同属 OpenAI 前端用户可见行为对齐工作，反映 frontend API 行为一致性治理的大方向。