PR #42752 完整报告

vllm-project/vllm

[Bugfix] Honor tool_choice="none" in Chat Completions streaming

合并时间：2026-06-04 04:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42752

执行摘要

- 一句话：修复流式 Chat Completions 中 tool_choice='none' 未生效
- 推荐动作：值得精读。此 PR 展示了流式与非流式路径一致性修复的典型模式，并体现了 review 过程中关于守卫位置和范围权衡的决策过程，有助于理解 vLLM 工具调用解析架构。

功能与动机

Issue #42747 报告：流式 Chat Completions 中 tool_choice='none' 未能阻止工具解析，只要服务器配置了 --tool-call-parser 且模型输出匹配格式，仍会生成 delta.tool_calls。非流式路径已正确处理，流式路径缺少等效守卫。

实现拆解

1. 在 _extract_tool_calls_streaming 中增加守卫（vllm/parser/abstract_parser.py:709）：在函数入口处检查 request.tool_choice == "none"，若成立则直接返回 (DeltaMessage(content=delta_text) if delta_text else None), False，避免调用底层 tool parser。
2. 简化 parse_delta：移除先前在 parse_delta 中临时添加的条件分支，恢复单一调用 _extract_tool_calls_streaming 的结构（由 reviewer sfeng33 建议，使所有 tool_choice 分支逻辑集中）。
3. 新增单元测试（tests/parser/test_streaming.py）：添加 test_parse_delta_tool_choice_none 和 test_parse_delta_tool_choice_none_with_reasoning，使用 Hermes2ProToolParser 模拟工具解析，断言当 tool_choice='none' 时，模型输出中的工具标记保留在 content 中且 tool_calls 为空。
4. 调整测试夹具：将 request_obj 的默认 tool_choice 改为 "auto" 并附加 tools 定义，确保测试回归仅由 tool_choice 覆盖触发。

关键文件：

- vllm/parser/abstract_parser.py（模块 解析器；类别 source；类型 core-logic；符号 _extract_tool_calls_streaming）：核心修复文件，在 _extract_tool_calls_streaming 增加 tool_choice == "none" 守卫，跳过工具解析。
- tests/parser/test_streaming.py（模块 测试；类别 test；类型 test-coverage；符号 test_parse_delta_tool_choice_none, test_parse_delta_tool_choice_none_with_reasoning）：新增两个单元测试，验证 tool_choice="none" 时工具解析被跳过，内容正确传递。同

时调整了测试夹具的默认 tool_choice 和 tools 参数。

关键符号：_extract_tool_calls_streaming, test_parse_delta_tool_choice_none, test_parse_delta_tool_choice_none_with_reasoning

关键源码片段

vllm/parser/abstract_parser.py

核心修复文件，在 _extract_tool_calls_streaming 增加 tool_choice == "none" 守卫，跳过工具解析。

```
# vllm/parser/abstract_parser.py - _extract_tool_calls_streaming ( 部分 )
def _extract_tool_calls_streaming(
    self,
    previous_text: str,
    current_text: str,
    delta_text: str,
    previous_token_ids: Sequence[int],
    current_token_ids: Sequence[int],
    delta_token_ids: Sequence[int],
    request: ChatCompletionRequest | ResponsesRequest,
    tool_call_idx: int | None = None,
    tool_call_id_type: str = "random",
    function_name_returned: bool = False,
) -> tuple[DeltaMessage | None, bool]:
    # 新增守卫：若 tool_choice 为 "none"，直接返回文本内容，跳过底层解析
    if request.tool_choice == "none":
        # 保留 delta_text 作为普通 content，返回 False 表示未触发 function_name
        return (DeltaMessage(content=delta_text) if delta_text else None), False

    assert self._tool_parser is not None
    supports_required_and_named = self._tool_parser.supports_required_and_named
    # 原有 required/named/auto 分支处理 ...
```

tests/parser/test_streaming.py

新增两个单元测试，验证 tool_choice="none" 时工具解析被跳过，内容正确传递。同时调整了测试夹具的默认 tool_choice 和 tools 参数。

```
# tests/parser/test_streaming.py - 新增测试
def test_parse_delta_tool_choice_none(tokenizer, request_obj):
    # 创建一个带 tool_parser 但不含 reasoning 的解析器
    parser = make_parser(tokenizer, reasoning=False, tool=True)
    # 覆盖 tool_choice 为 "none"
    request = request_obj.model_copy(update={"tool_choice": "none"})
    # 流式处理 MODEL_OUTPUT（包含 <tool_call>...</tool_call> 标记）
    results = stream_text(parser, tokenizer, MODEL_OUTPUT, request, prompt_token_ids=[])
    reasoning, content, tool_calls = collect_fields(results)

    assert reasoning == ""
```

```
    assert len(tool_calls) == 0 # 确保未生成工具调用
    assert "<tool_call>" in content # 原始标记保留在 content 中
    assert "get_weather" in content


def test_parse_delta_tool_choice_none_with_reasoning(tokenizer, request_obj):
    # 同时启用 reasoning 和 tool_parser
    parser = make_parser(tokenizer, reasoning=True, tool=True)
    request = request_obj.model_copy(update={"tool_choice": "none"})
    results = stream_text(parser, tokenizer, MODEL_OUTPUT, request, prompt_token_ids=[])
    reasoning, content, tool_calls = collect_fields(results)

    assert "let me think about this" in reasoning # reasoning 正常提取
    assert len(tool_calls) == 0
    assert "<tool_call>" in content
    assert "get_weather" in content
```

评论区精华

守卫位置争论：reviewer sfeng33 要求将守卫从 parse_delta 移动到 _extract_tool_calls_streaming，因为后者已集中处理 required/named 等 tool_choice 分支，避免 parse_delta 增加额外条件。提交者 hoobnn 采纳并调整了返回逻辑以保留文本内容。

守卫范围讨论：gemini-code-assist[bot] 和 depthfirst-app[bot] 均建议同时覆盖 request.tool_choice is None 以保证与非流式完全一致。但最终 sfeng33 将条件缩窄为 == "none"，理由是 None 应按照 OpenAI 规范视为 auto。PR body 中 hoobnn 曾折叠 #44102 的放宽版本，但合并时采纳了 maintainer 的决策。

测试覆盖建议：reviewer mychmly 建议增加 ResponsesRequest(tool_choice="none") 的回归测试，hoobnn 表示已添加。最终版本至少包含两个单元测试。

- 守卫位置：parse_delta vs _extract_tool_calls_streaming (design): hoobnn 采纳建议，将守卫移至 _extract_tool_calls_streaming 顶部，并调整返回逻辑以保留文本内容。
- 守卫范围：是否应包含 request.tool_choice is None (correctness): 最终合并版本仅检查 == "none"，None 保持 auto 行为。
- 测试覆盖建议：ResponsesRequest 路径 (testing): hoobnn 表示已添加，但最终提交仅包含 ChatCompletion 测试，可能后续补充。

风险与影响

- 风险：低风险。改动仅 3 行源码（在 _extract_tool_calls_streaming 入口增加早期返回），且经过单元测试验证。唯一潜在问题是：若用户期望 tool_choice: null 禁止工具但服务将其视为 auto，则可能仍产生意外工具调用。但此行为符合 OpenAI 规范，且非流式路径已如此处理。
- 影响：用户影响：修复了流式 Chat Completions 中 tool_choice="none" 行为与非流式不一致的bug；使用--tool-call-parser的用户将不再在none模式下收到错误delta.tool_calls。 系统影响：仅修改解析器模块，不影响性能或稳定性。 团队影响：约 40 行新增代码，容

易理解和维护。

- 风险标记：核心解析路径变更，测试未覆盖 null 场景（设计决策）

关联脉络

- PR #44102 [BugFix] Honor tool_choice="none" in Chat Completions streaming (broader guard): 独立实现的相同修复，最初包含 None 守卫；该 PR 的 review 方向被折叠到本 PR，后因 maintainer 决策被缩小范围。PR body 中致谢并标记为'covered'。
- PR #42868 Unrelated approach: patch chat_completion/serving.py: 另一条修复尝试，但修改位置不同；本 PR 采用更集中的解析层修改。
- PR #42747 [Bug]: Chat Completions streaming invokes tool parser despite tool_choice="none": 直接关联的 issue，描述 bug 及复现步骤。
- PR #9776 [BugFix] Honor tool_choice="none" in Chat Completions streaming (vllm-ascend mirror): 本 PR 在 vllm-ascend 的镜像修复，引用同一 issue 和方案。