

PR #48922 完整报告

vllm-project/vllm

[Bugfix] Guard tool call argument JSON parsing in chat message postprocessing

合并时间: 2026-08-27 04:26

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48922>

执行摘要

- 一句话: 修复 tool call 参数 JSON 解析失败导致会话不可恢复
- 推荐动作: 值得精读。核心看两处: 一是在历史消息重放场景下 coercion 与 400 的权衡论证 (PR body 的三点理由与 bbrowning 的最终裁决); 二是以 gemma4 jinja 为例说明 chat template 对 mapping 的硬约束如何反过来决定入口层的容错策略。测试对边界的系统覆盖 (畸形 / 非对象 / 缺失 / 多条调用独立收敛) 也值得参考。

功能与动机

issue #47761 报告 Qwen3.6-35B-A3B-FP8 在生成代码时返回 400 'Unterminated string starting at'。PR body 指出: 两个独立触发点 (Generation 撞 max_tokens 中途截断、Responses 路径 EOS 时字符串未闭合) 汇聚到同一个无保护的 `json.loads`; 坏值位于对话历史中, 客户端每轮重放同一条 assistant 消息都会复现同一错误, 会话只能丢弃。PR 作者排除了 tool parser、客户端转义、chat template 配置和版本回归等假设, 并验证 payload 与 SSE delta 重建结果字节一致 (md5 相同), 因此修复必须落在 `_postprocess_messages` 这个唯一汇聚点。

实现拆解

1. 变更入口: `vllm/entrypoints/chat_utils.py` 的 `_postprocess_messages`, 它是 Chat Completions 与 Responses 两条路由对 assistant 消息后处理的汇聚点; 本 PR 只改它内部 `tool_calls` 循环中对 `function["arguments"]` 的处理分支 (+37 / -3)。
2. 核心逻辑重构: 把原来的单分支 `if not isinstance(content, (dict, list)): parsed = json.loads(content)` 拆成类型分派——dict 直接透传; str 用 `json.loads` 解析, 捕获 `json.JSONDecodeError` 后记录 warning (含工具名与 payload 长度, 不记录 payload 内容) 并将 `parsed` 置为 `None`; 其他类型 (如 list) 原样保留到统一校验。随后统一校验: `parsed` 非 dict 时, 若非 `None` 再发一条告警说明解码结果类型, 最终都置为 `{}`; `null` 解析为 `None`, 按既有语义直接变 `{}` 且不告警。缺失、空字符串、"null" 依旧走 `function["arguments"] = {}` 的旧路径。
3. 设计约束: 所有值只允许 dict 进入 chat template。PR body 和 commit 历史显示早期版本曾把原始字符串透传, review 中发现 `examples/tool_chat_template_gemma4.jinja` 会对字符串显式 `raise_exception`、`toolace` 与 `pythonic` 模板抛 `TypeError`、`gemma3_pythonic` 逐字符迭代, 因此移除透传。

4. 测试配套: tests/entrypoints/unit_tests/test_chat_utils.py 新增 117 行, 辅助函数 `_assistant_tool_call / _assistant_tool_calls` 构造单条 / 多条 tool call 消息; 7 组测试覆盖 dict 直通、合法对象字符串、小型畸形 JSON (复刻 @iMMIQ 的 54 字节 payload)、大型畸形 JSON (复刻 18k 字符 payload)、合法非对象 JSON (参数化 `[]/42/true/"hello"/null`)、缺失或空参数、多 tool call 相互独立; 用 caplog 断言警告内容。
5. 协作与演进: 4 个 commit 包含两次 main 合并; 评审阶段从「透传原始字符串」改为「仅 dict 进入模板」; 最终由 bbrowning 批准合并。

关键文件:

- vllm/entrypoints/chat_utils.py (模块 请求入口; 类别 source; 类型 core-logic; 符号 `_postprocess_messages`): 核心修复文件: `_postprocess_messages` 对 `tool_calls[].function.arguments` 增加防御式 JSON 解析, 畸形或非对象的 JSON 收敛为 `{}` 并告警, 仅 dict 进入 chat template。该函数是所有 Chat Completions / Responses 请求的后处理汇聚点。
- tests/entrypoints/unit_tests/test_chat_utils.py (模块 单元测试; 类别 test; 类型 test-coverage; 符号 `_assistant_tool_call, _assistant_tool_calls, test_tool_call_arguments_dict_passthrough, test_tool_call_arguments_valid_object_string`): 新增 117 行测试, 构造单条 / 多条 tool call 消息, 覆盖 dict 直通、合法对象字符串、小 / 大畸形 JSON、合法非对象 JSON、缺失或空参数、多 tool call 独立收敛, 用 caplog 断言告警内容。

关键符号: `_postprocess_messages, _assistant_tool_call, _assistant_tool_calls, test_tool_call_arguments_dict_passthrough, test_tool_call_arguments_valid_object_string, test_tool_call_arguments_malformed_json_small, test_tool_call_arguments_malformed_json_large, test_tool_call_arguments_valid_json_non_object, test_tool_call_arguments_missing_or_empty`

关键源码片段

vllm/entrypoints/chat_utils.py

核心修复文件: `_postprocess_messages` 对 `tool_calls[].function.arguments` 增加防御式 JSON 解析, 畸形或非对象的 JSON 收敛为 `{}` 并告警, 仅 dict 进入 chat template。该函数是所有 Chat Completions / Responses 请求的后处理汇聚点。

```
# vllm/entrypoints/chat_utils.py
# _postprocess_messages 中处理 assistant tool_calls 的核心循环体。
# 背景: #47761 中 vLLM 生成被截断的 arguments 字符串后, 客户端在后续
# 每一轮都会重放这条历史消息; 若这里直接抛错, 会话将永远不可恢复,
# 因此本分支的目标是: 只有 dict 允许进入 chat template, 其余一律收敛。
```

```
for item in tool_calls:
    if not isinstance(item, dict):
        raise VLLMValidationError(
            "assistant tool_calls entries must be objects.",
            parameter="tool_calls",
```

```

    )

function = item.get("function")
if item.get("type", "function") != "function" or not isinstance(
    function, dict
):
    raise VLLMValidationError(
        "chat completions only support assistant tool_calls "
        "of type 'function'.",
        parameter="tool_calls",
    )

# arguments 缺失、空字符串或 None 时保持原有行为：置为 {}
if content := function.get("arguments"):
    if isinstance(content, dict):
        # dict 直接透传，无需 JSON 解析
        parsed = content
    else:
        if isinstance(content, str):
            try:
                parsed = json.loads(content)
            except json.JSONDecodeError:
                # 畸形 JSON 不能抛错：该值在对话历史里，抛错会让
                # 后续每一轮都失败。只记录工具名与 payload 长度，
                # 不记录 payload 本身（可能包含用户源码）。
                logger.warning(
                    "Tool call %r has arguments that are not valid "
                    "JSON (%d chars); coercing to an empty object "
                    "so the conversation can continue.",
                    function.get("name"),
                    len(content),
                )
                parsed = None
        else:
            # 非 str 非 dict（如 list），交给下方统一判断
            parsed = content

    if not isinstance(parsed, dict):
        if parsed is not None:
            # 合法 JSON 但不是对象（如 "[]"、"42"、"true"），
            # chat template 要求 mapping，同样收敛为 {}。
            logger.warning(
                "Tool call %r arguments decoded to %s, not a "
                "JSON object; coercing to an empty object.",
                function.get("name"),
                type(parsed).__name__,
            )
        parsed = {}

```

```

        function["arguments"] = parsed
    else:
        function["arguments"] = {}

```

tests/entrypoints/unit_tests/test_chat_utils.py

新增 117 行测试，构造单条 / 多条 tool call 消息，覆盖 dict 直通、合法对象字符串、小 / 大畸形 JSON、合法非对象 JSON、缺失或空参数、多 tool call 独立收敛，用 caplog 断言告警内容。

```

# tests/entrypoints/unit_tests/test_chat_utils.py
# 辅助构造函数：生成一条带单个 function tool_call 的 assistant 消息，
# 让测试用例聚焦于 arguments 的解析行为。

```

```

def _assistant_tool_call(arguments, name="write"):
    return [
        {
            "role": "assistant",
            "content": "",
            "tool_calls": [
                {
                    "id": "call_0",
                    "type": "function",
                    "function": {"name": name, "arguments": arguments},
                }
            ],
        }
    ]

```

```

def test_tool_call_arguments_malformed_json_small(caplog):
    # 对应 #47761 中 EOS 截断的 54 字节 payload（未闭合引号），
    # 期望收敛为 {} 且只产生一条包含工具名与收敛提示的警告。
    bad = '{"cmd": "mkdir -p /tmp/mmadtest && cat > /tmp/mmadtest'
    messages = _assistant_tool_call(bad, name="exec")
    _postprocess_messages(messages)
    args = messages[0]["tool_calls"][0]["function"]["arguments"]
    assert args == {}

    assert len(caplog.records) == 1
    assert "exec" in caplog.records[0].message
    assert "coercing to an empty object" in caplog.records[0].message

```

```

@pytest.mark.parametrize("non_obj", ["[]", "42", "true", '"hello"', "null"])
def test_tool_call_arguments_valid_json_non_object(caplog, non_obj):
    # 合法 JSON 但不是对象：除 null（解析为 None）外都应告警并收敛
    messages = _assistant_tool_call(non_obj, name="bad_tool")
    _postprocess_messages(messages)
    args = messages[0]["tool_calls"][0]["function"]["arguments"]

```

```

assert args == {}

if non_obj == "null":
    # null 解析为 None, 按需求不产生警告
    assert not caplog.records
else:
    assert len(caplog.records) == 1
    assert "bad_tool" in caplog.records[0].message
    assert "not a JSON object" in caplog.records[0].message

```

评论区精华

bbrowning 先给出 CHANGES_REQUESTED: 不能把非 dict JSON 透传给 chat template, 模板普遍期望 JSON dict, 并以 [examples/tool_chat_template_gemma4.jinja](#) 显式 raise 为例。VBS2004 确认该模板的 [raise_exception](#) 后删除透传分支, 只允许 dict 到达模板。随后 bbrowning 在 APPROVED 中明确: 考虑到多轮场景, coercion 优于 400——若 vLLM 自己产出的坏参数在下一轮导致 400, 这是不可恢复的问题; coercion 至少让模型有机会反应并纠正。

- 非 dict JSON 值能否透传给 chat template (design): VBS2004 确认 gemma4 模板证据后删除透传分支, 只允许 dict 到达模板; 非 dict 一律告警并收敛为 {}。
- 畸形 JSON 时 coercion 与 400 的选择 (design): bbrowning 最终同意 coercion: vLLM 自己产出的坏参数若导致下一轮 400 是不可恢复问题; coercion 至少给模型机会反应并纠正。

风险与影响

- 风险:
 1. 行为变更风险: 畸形 / 非对象 arguments 从 400 变为静默 {}, 工具调用参数被丢弃, 依赖 400 语义的客户端 (如超时重试、熔断、前端错误提示) 行为会变化, 需在 release note 中说明。
 2. 数据质量风险: 告警日志是唯一信号, 但并非所有部署都会采集 warning; 上游根因 (finish_reason 应报 "length"、Responses Simple 路径发送前校验) 未在本 PR 修复, 坏参数仍可能继续产生, 本修复只是兜底。
 3. 一致性风险: 若 #50861 按 400 方案合并, 两条路径对同一个畸形输入会给出不同行为 (400 vs {}), 可能造成调试困惑。
 4. 性能影响: 正常路径仅多几次 isinstance 判断; 异常路径的 json.loads 失败在超长字符串上代价可忽略。
 5. 兼容性: dict 直通与合法对象字符串解析为 dict 的行为不变, {"a": 1} 与 '{"a": 1}' 结果一致, 原有 "null" → {} 语义保留。- 影响: 所有使用 Chat Completions / Responses 且重放 assistant tool_calls 的多轮会话 (工具调用、代码生成等场景) 都会经过 _postprocess_messages 此分支, 影响面广但分支逻辑简单。对受影响的用户, 会话从「必须丢弃」变为「可继续并让模型纠错」; 对团队, 确立了「不可用 arguments 一律收敛 {}」的入口容错策略, 为同函数系列 PR (#43862 / #47555 / #50861) 提供统一的参照基线。- 风险标记: 核心请求路径变更, 静默数据收敛, 行为与 #50861 潜在冲突, 上游根因未修复

关联脉络

- PR #50861 [Related] Alternative approach: raise VLLMValidationError on malformed arguments: PR body 与 review 中反复对比的替代方案：畸形 JSON 时返回 400；本 PR 最终选择 coercion，两者仅在畸形分支行为不同。
- PR #43862 [Related] Merged precedent: coerce arguments "null" to {} in `_postprocess_messages`: 同函数 `_postprocess_messages` 中已有 coercion 先例，本 PR 将其扩展为「不可用 arguments 一律收敛 {}」的通用策略。
- PR #47555 [Related] Extend coercion to scalar arguments in `_postprocess_messages`: 同函数系列 PR，将 coercion 覆盖到标量；本 PR 补齐「畸形 JSON」这一类。
- PR #53854 [Bugfix][Processor] Replace bare asserts with ValueError in DeepseekVLV2/OCR processors: 同属「把输入校验从崩溃 /400 变成可诊断、可恢复」的入口健壮性趋势，但模块不同，可作为横向前后对照。