PR #48171 完整报告

vllm-project/vllm

[Bugfix] Fix lfm2 tool parser dropping calls with brackets or newline…

合并时间：2026-08-10 15:10

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48171

执行摘要

- 一句话：修复 lfm2 工具解析器丢弃含括号换行等参数的工具调用
- 推荐动作：值得精读，重点看 vllm/tool_parsers/utils.py 中 quote-aware 扫描器与候选 first-wins 设计，以及控制符转义限定在 lfm2 调用点的共享边界决策，对任何想加固 LLM 输出解析的团队都有参考价值。合并后建议观察 lfm2 用户对流式首 token 延迟和 tuple / set 参数输出形状的反馈。

功能与动机

PR body 指出 lfm2 解析器 silently drops (or corrupts) tool calls for a range of outputs that real agentic models emit routinely。真实 agentic traces（SWE-agent / OpenClaw 风格 shell 命令）频繁命中 ast.parse 失败、括号栈错乱或 unsupported AST 节点；其中 commit 9020bd 提到对约 5.9M 条记录的扫描发现 30,217 次工具调用因负数参数被静默丢弃（单个数据集最高占 34.6% 行）。PR 的目标是把可恢复的输出全部恢复，且所有改写对已合法输入是 no-op、恢复出的参数值能精确 round-trip 回原始控制字符。

实现拆解

1. 扩展共享字面量提取（vllm/tool_parsers/utils.py 的 get_parameter_value）：新增 ast.Tuple / ast.Set 按源码顺序转 list、无占位符 JoinedStr（f'hello'）折叠为 str、UnaryOp(+/-) 数值取反分支；同时把 ast.Constant 分支收窄为 JSON 可表示标量，bytes / Ellipsis / complex 就地抛 UnexpectedAstError，避免在 json.dumps 深处以 TypeError 级联丢弃整个调用列表。这些扩展对其他 pythonic parser（pythonic / llama4_pythonic / olmo3）一并生效。
2. 新增 quote-aware 文本改写工具集（utils.py）：escape_ctrl_chars_in_strings 只转义字符串字面量内的换行 / CR / Tab / NUL；rename_reserved_kwargs 与 restore_reserved_kwarg_names 只在关键字参数位置（前置 (或 ,，后接单个 =）把保留字改名再解码后还原；normalize_leading_zero_ints 只剥离小数整数字面量前导零；escape_nested_quotes_in_strings 配合 unescaped_quotes / is_closer / contains_broken_string_literal 对歧义嵌套引号做候选闭合恢复。每个函数都用 quote 状态机避免触达字符串外文本，对已合法文本完全无操作。
3. 接入 lfm2 非流式 extract_tool_calls：先直接 ast.parse，失败后构造候选序列（控制符转义 + 前导零剥离 → 嵌套引号恢复后再转义 → 保留字改名），第一个能 parse 的候选胜出；kw_renamed 时用新增的 _restore_reserved 解码 JSON 并还原参数名；空块 [] 通过

parsed.elts 非空检查，不再返回 tools_called=True 的零调用结果。

4. 接入 lfm2 流式 extract_tool_calls_streaming：同样先 escape / normalize，但嵌套引号恢复只在 has_end_in_current 且文本仍无法 parse 时进行，保证跨 chunk 判断稳定；contains_broken_string_literal 时 withhold delta 直到恢复产出稳定文本；并修复 sentinel 后空白导致的 IndentationError（tool_text.lstrip）。
5. 配套测试：tests/tool_parsers/test_utils.py 新增 485 行 utils 级回归（每个失败类有最小复现与 round-trip 断言，并用 _value_of / _first_call / _bare_call / _kwarg_constant 辅助通过 mypy 类型收窄）；tests/tool_parsers/test_lfm2_tool_parser.py 新增 198 行 e2e 用例（约 20 个新场景 × 流式与非流式，外加空块与 sentinel 后空白边界）。

关键文件：

- vllm/tool_parsers/utils.py（模块 解析工具；类别 source；类型 core-logic；符号 get_parameter_value, escape_ctrl_chars_in_strings, rename_reserved_kwargs, restore_reserved_kwarg_names）：核心实现地：扩展 get_parameter_value 支持负数 / tuple / set / 无占位符 f-string 并显式拒绝非 JSON 常量，新增 5 个 quote-aware 文本改写函数，构成全部恢复逻辑的底层能力。
- vllm/tool_parsers/lfm2_tool_parser.py（模块 解析器；类别 source；类型 core-logic；符号 extract_tool_calls, extract_tool_calls_streaming, _restore_reserved）：恢复逻辑真正生效的接入点：非流式与流式路径的渐进式改写链、保留字参数名还原、空块与 sentinel 后空白修复。
- tests/tool_parsers/test_utils.py（模块 解析测试；类别 test；类型 test-coverage；符号 _value_of, _first_call, _bare_call, _kwarg_constant）：每个失败类的最小回归，覆盖 get_parameter_value / make_valid_python / 各改写函数的行为与 round-trip 保真，并用类型收窄辅助通过 mypy。
- tests/tool_parsers/test_lfm2_tool_parser.py（模块 解析测试；类别 test；类型 test-coverage；符号 test_empty_tool_call_block, test_whitespace_after_start_token, test_tool_call）：e2e 流式与非流式用例，外加空块和 sentinel 后空白边界，验证恢复在真实 parser 路径生效。

关键符号：get_parameter_value, escape_ctrl_chars_in_strings, rename_reserved_kwargs, restore_reserved_kwarg_names, normalize_leading_zero_ints, escape_nested_quotes_in_strings, contains_broken_string_literal, unescaped_quotes, is_closer, extract_tool_calls, extract_tool_calls_streaming, _restore_reserved

关键源码片段

vllm/tool_parsers/utils.py

核心实现地：扩展 get_parameter_value 支持负数 / tuple / set / 无占位符 f-string 并显式拒绝非 JSON 常量，新增 5 个 quote-aware 文本改写函数，构成全部恢复逻辑的底层能力。

```
def get_parameter_value(val: ast.expr) -> Any:
    """从 AST 表达式节点提取工具调用参数的 Python 字面量值。

    本 PR 扩展了负数、tuple、set、无占位符 f-string 的识别，
```

```
并显式拒绝 JSON 不可表示的常量，避免其在 json.dumps 深处
以 TypeError 级联丢弃整个调用列表。
"""
if isinstance(val, ast.Constant):
    # 仅接受 JSON 可表示的标量；bytes / Ellipsis / complex 之前
    # 能通过本函数，却在 json.dumps 里抛 TypeError，导致同列表的
    # 其他工具调用一并被丢弃，现在就地报 UnexpectedAstError。
    if val.value is None or isinstance(val.value, (str, int, float)):
        return val.value
    logger.warning('Non-JSON-representable constant in tool call arguments')
    raise UnexpectedAstError('Tool call arguments must be JSON values')
elif isinstance(val, ast.Dict):
    # dict 键必须是常量，值递归提取，保持原有行为。
    if not all(isinstance(k, ast.Constant) for k in val.keys):
        logger.warning('Dict argument keys are not all literals')
        raise UnexpectedAstError('Dict tool call arguments must have literal keys')
    return {k.value: get_parameter_value(v) for k, v in zip(val.keys, val.values)}
elif isinstance(val, ast.List):
    return [get_parameter_value(v) for v in val.elts]
elif isinstance(val, ast.Tuple):
    # JSON 没有 tuple 类型，size=(800, 600) 按 list 解码以能
    # 通过 json.dumps；不处理则整个调用被丢弃。
    return [get_parameter_value(v) for v in val.elts]
elif isinstance(val, ast.Set):
    # JSON 没有 set 类型，tags={'urgent', 'bug'} 按源码顺序转 list。
    return [get_parameter_value(v) for v in val.elts]
elif isinstance(val, ast.JoinedStr) and all(
    isinstance(part, ast.Constant) for part in val.values
):
    # 无占位符 f-string（f'hello'）在 AST 中是 JoinedStr 而非
    # Constant；折叠成普通字符串。带真实占位符的 f-string 仍
    # 落入下方拒绝分支。
    return ''.join(str(part.value) for part in val.values)
elif isinstance(val, ast.Name) and val.id in _JSON_NAME_LITERALS:
    return _JSON_NAME_LITERALS[val.id]
elif isinstance(val, ast.UnaryOp) and isinstance(val.op, (ast.USub, ast.UAdd)):
    # Python 把 -1 解析为 UnaryOp(USub, Constant(1)) 而非普通
    # 常量，负纬度、负偏移等参数因此被整批丢弃（扫描 5.9M 条
    # 记录发现 3 万余例）。这里只对数值操作数做取反，保证
    # not / ~ 等真表达式仍然被拒绝。
    operand = get_parameter_value(val.operand)
    if isinstance(operand, (int, float)) and not isinstance(operand, bool):
        return -operand if isinstance(val.op, ast.USub) else operand
    logger.warning('Unsupported unary operand in tool call arguments')
    raise UnexpectedAstError('Tool call arguments must be literals')
else:
    logger.warning('Unsupported AST node type in tool call arguments')
    raise UnexpectedAstError('Tool call arguments must be literals')
```

vllm/tool_parsers/lfm2_tool_parser.py

恢复逻辑真正生效的接入点：非流式与流式路径的渐进式改写链、保留字参数名还原、空块与 sentinel 后空白修复。

```
    @staticmethod
    def _restore_reserved(tool_call):
        """还原 rename_reserved_kwargs 改名的保留字参数名。

        解析器在 AST 层把 from=1 改写为 from_pyreservedkw_=1 才能通过
        ast.parse；解码 JSON arguments 后把带后缀且词干确为 Python
        关键字的键还原为原名。
        """
        arguments = json.loads(tool_call.function.arguments)
        restored = restore_reserved_kwarg_names(arguments)
        if restored != arguments:
            tool_call.function.arguments = json.dumps(restored, ensure_ascii=False)
        return tool_call

    # 非流式路径：先直解，失败时按序尝试渐进式改写候选。
    try:
        kw_renamed = False
        try:
            module = ast.parse(tool_text)
        except (SyntaxError, ValueError):
            # 每条改写对已合法文本都是 no-op。顺序：先转义字符串内
            # 控制字符并剥离整数字面量前导零；再尝试嵌套引号恢复
            # （可能引入新的控制字符，故再次转义）；最后改名保留字
            # 参数。第一个能通过 ast.parse 的候选胜出。
            escaped = escape_ctrl_chars_in_strings(
                normalize_leading_zero_ints(tool_text)
            )
            candidates = [escaped]
            requoted, requote_changed = escape_nested_quotes_in_strings(escaped)
            if requote_changed:
                candidates.append(escape_ctrl_chars_in_strings(requoted))
            renamed, kw_renamed = rename_reserved_kwargs(candidates[-1])
            if kw_renamed:
                candidates.append(renamed)
            for candidate in candidates:
                try:
                    module = ast.parse(candidate)
                    break
                except (SyntaxError, ValueError):
                    continue
            else:
                raise
        parsed = getattr(module.body[0], 'value', None)
        # 空块 [] 必须与“调用了一个工具”区分：要求至少一个元素，
```

```
            # 否则会返回 tools_called=True 但 tool_calls 为空。
            if (
                isinstance(parsed, ast.List)
                and parsed.elts
                and all(isinstance(e, ast.Call) for e in parsed.elts)
            ):
                tool_calls = [handle_single_tool(e) for e in parsed.elts]
                if kw_renamed:
                    tool_calls = [self._restore_reserved(tc) for tc in tool_calls]
                return ExtractedToolCallInformation(
                    tools_called=True, tool_calls=tool_calls, content=content
                )
            raise UnexpectedAstError('Tool output must be a list of function calls')
        except Exception:
            logger.exception('Error in extracting tool call from response.')
            return ExtractedToolCallInformation(
                tools_called=False, tool_calls=[], content=model_output
            )
```

评论区精华

该 PR 没有实际行内 review 评论（review_comments_count=0）。claude[bot] 指出 PR 来自 fork，自动审查已禁用，维护者可手动触发一次性审查；最终由合并者 chaunceyjiang 单审 APPROVED，评论仅为 LGTM。PR body 还主动说明两点边界：与 #46708 触碰同一函数但修复不同 edge case（闭引号前的反斜杠序列计数），确认无重叠；控制符转义只接入 lfm2 call site，确保 pythonic / llama4_pythonic / olmo3 的流式行为不变。

- fork PR 自动审查被禁用 (other): 未触发 bot 审查；由维护者 chaunceyjiang 手动审查并 APPROVED，评论仅为 LGTM。
- 与 #46708 的边界划分 (question): 确认无重叠、共享语义保持。

风险与影响

- 风险：
  1. 共享行为放宽：utils.py 的 get_parameter_value 扩展同时影响 pythonic / llama4_pythonic / olmo3，tuple / set / f-string / 负数在这些 parser 中也会被接受并按 list / 数值输出，输出形状可能变化（如 tuple 变 list），属于行为面扩张，虽有 60 个既有回归测试背书，但用户可感知结果可能不同。
  2. 流式改写跨 chunk 一致性：流式路径依赖改写函数的确定性，contains_broken_string_literal 若对合法嵌套引号误判，会不必要 withhold delta，增加首 token 延迟或造成流式间断。
  3. 嵌套引号恢复是启发式：escape_nested_quotes_in_strings 对极端引号组合可能选择非预期闭合点，依赖 ast-validated 候选兜底；候选链 first-wins 策略理论上可能让某种输入以非预期取值恢复，但至少不再丢调用。
  4. 从提交历史看共有 5 个功能 commit 带有 Claude co-authored 署名，且 6 次 merge main；无实际 review 讨论记录，合并决策主要由测试与 CI 背书。 - 影响：影响使用

--tool-call-parser lfm2 的 agentic 场景：shell 命令、坐标、保留字参数等真实输出不再被静默丢弃或被截断，流式与非流式行为对齐。共享 get_parameter_value 的扩展会轻微放宽 pythonic / llama4_pythonic / olmo3 的输入接受面，但对应测试套件全部通过。总改动约 +1186/-24，其中源码约 500 行、测试约 683 行，无文档、配置或部署配套变更。 - 风险标记：共享工具函数行为变更，流式改写依赖确定性，嵌套引号启发式风险，解析器接受面放宽

关联脉络

- PR #46708 PR #46708（标题未在上下文中提供）：PR body 声明与它修同一函数 make_valid_python 的不同 edge case（反斜杠序列计数），为相邻改动线，确认无重叠。
- PR #50528 [Bugfix][Parser] Emit REASONING_END for Inkling tool calls that follow no thinking block: 同为 tool-calling parser 的流式解析边界修复，体现 parser 健壮性演进的同类工作。
- PR #49876 [Bugfix][Parser] Confirm reasoning end when an Inkling content block opens: 同样针对无思考块时的工具调用流式解析边界修复，与本次恢复性解析目标一致。
- PR #51391 [Bugfix][Parser] Prevent Inkling block-end leakage with tools: 同一解析器家族中针对流式块结束泄漏的修复，共享回归测试基础设施。