PR #50515 完整报告

vllm-project/vllm

[ROCm][CI] Restore Mistral tool-parser compatibility after unification

合并时间：2026-08-01 00:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50515

执行摘要

- 一句话：修复 Mistral 工具解析器统一后的兼容性回归
- 推荐动作：值得快速精读。这是统一重构后兼容性适配的典型修复样例：通过 adapter 属性转发恢复嵌套 engine 的对外契约，并通过显式 None 分支保留旧路径的可空 request 语义。评审中「getattr 兜底 vs 显式 None 检查 + 类型标注同步修改」的取舍值得借鉴。若后续有同类 parser 统一工作，应参考本 PR 补充针对 request=None 与属性转发的单元测试。

功能与动机

PR body 通过 git bisect 定位到 PR #48947 是首次引入该失败的修订：统一 parser 将 bot_token 和 bot_token_id 移到了嵌套 engine 上，而既有调用方仍在适配器上访问它们；同时 _legacy_extract_tool_calls 直接访问 request.tool_choice 和 request.tools，但既有 parser API 允许 request=None 且旧实现容忍这一点。回归由 AMD CI build 11504 的 Language Models Test (Extended Generation) 暴露。

实现拆解

1. 恢复适配器属性暴露（vllm/tool_parsers/mistral_tool_parser.py）：在 MistralToolParser 上以 @property 形式重新定义 bot_token 与 bot_token_id，直接转发到 self._parser_engine，使统一 parser 后仍可在适配器层读取 Mistral 工具调用标记及其 token id，恢复旧 MistralToolParser 的对外契约。
2. 放宽 _legacy_extract_tool_calls 的 request 可空性（vllm/parser/mistral.py）：将参数类型从 ChatCompletionRequest 改为 ChatCompletionRequest | None，在函数开头显式分支解包 tool_choice 与 tools 为局部变量，后续判断统一改用局部变量，避免 request 为 None 时访问属性抛 AttributeError。
3. 配套验证：本 PR 未新增单元测试，回归验证依赖 AMD ROCm CI 的 Extended Generation 测试；review 中评审者建议放弃 getattr 兜底写法、改为显式 None 检查并同步修改类型标注，作者已在第二个 commit 中采纳。

关键文件：

- vllm/tool_parsers/mistral_tool_parser.py（模块 工具解析；类别 source；类型 core-logic；符号 bot_token, bot_token_id）：在统一 parser 的适配器层恢复 bot_token / bot_token_id 属性并转发至嵌套 engine，是修复调用方属性访问失败的关键入口。

- vllm/parser/mistral.py（模块 解析引擎；类别 source；类型 core-logic；符号 _legacy_extract_tool_calls）：_legacy_extract_tool_calls 放宽为可空 request 并显式处理 None 分支，是恢复旧解析器对 request=None 容忍度的核心逻辑改动。

关键符号：bot_token, bot_token_id, _legacy_extract_tool_calls, extract_tool_calls_from_content

关键源码片段

vllm/tool_parsers/mistral_tool_parser.py

在统一 parser 的适配器层恢复 bot_token / bot_token_id 属性并转发至嵌套 engine，是修复调用方属性访问失败的关键入口。

```
class MistralToolParser(MistralParserToolAdapter):
    # 统一 parser 重构后，bot_token / bot_token_id 只存在于嵌套的 parser engine 上，
    # 而既有调用方仍在适配器层访问这两个属性，因此必须显式转发以恢复旧行为。
    IS_MISTRAL_TOOL_PARSER = True

    # Mistral 以 [TOOL_CALLS]name[ARGS]{...} 格式输出工具调用，
    # 与 serving 层通用 required / named handler 期望的 JSON 数组格式不同，
    # 因此把 required / named 也路由到本解析器的提取逻辑（按 auto 处理），
    # 避免解析崩溃或把原始信封直接灌进 arguments。
    supports_required_and_named = False

    @property
    def bot_token(self) -> str:
        """返回旧版解析器暴露的 Mistral 工具调用标记。"""
        return self._parser_engine.bot_token

    @property
    def bot_token_id(self) -> int | None:
        """返回 bot_token 对应的 token id；tokenizer 未提供时为 None。"""
        return self._parser_engine.bot_token_id

    def adjust_request(
        self,
        request: ChatCompletionRequest | ResponsesRequest,
    ) -> ChatCompletionRequest | ResponsesRequest:
        # 跳过基类 tool_choice -> structured_outputs 转换：
        # Mistral 通过 grammar mode（auto / none / required / named）强制 tool_choice，
        # 若把 tool 派生的 json_schema 并入 grammar，会允许裸 JSON 混入，
        # 破坏真正的 [TOOL_CALLS] 调用格式。用户显式提供的结构化输出
        # 仍由 engine 的 adjust_request 处理。
        return self._parser_engine.adjust_request(request)
```

vllm/parser/mistral.py

_legacy_extract_tool_calls 放宽为可空 request 并显式处理 None 分支，是恢复旧解析器对 request=None 容忍度的核心逻辑改动。

```
def _legacy_extract_tool_calls(
    self,
    model_output: str,
    request: ChatCompletionRequest | None,
) -> ExtractedToolCallInformation:
    """Pre-v11 非流式提取，处理 [TOOL_CALLS] 信封与受引导的裸数组格式。

    统一 parser 后调用方可能以 request=None 进入该路径（旧 MistralToolParser
    一直容忍这种用法），因此先显式解包，而不是直接访问
    request.tool_choice / request.tools 导致 AttributeError。
    """
    if request is None:
        tool_choice = None
        tools = None
    else:
        tool_choice = request.tool_choice
        tools = request.tools

    # tool_choice 为 "none" 且同时给出 tools 时，绝不产生工具调用。
    if tool_choice == "none" and tools:
        return ExtractedToolCallInformation(
            tools_called=False, tool_calls=[], content=model_output
        )

    content: str | None = None
    if self.bot_token in model_output:
        content_and_raw_tool_calls = model_output.split(self.bot_token)
        content = content_and_raw_tool_calls[0]
        raw_tool_calls = content_and_raw_tool_calls[1:]
        # pre-v11 格式：content[BOT][{tool_call1},{tool_call2}]
        if len(raw_tool_calls) != 1:
            raise ValueError(
                "Only one BOT token should have been outputted, "
                f"but got {model_output}."
            )
        stringified_tool_calls = raw_tool_calls[0].strip()
    elif tool_choice == "required" or isinstance(
        tool_choice, ChatCompletionNamedToolChoiceParam
    ):
        # 受引导的裸数组输出（没有 [TOOL_CALLS] 标记）。
        stringified_tool_calls = model_output.strip()
    else:
        return ExtractedToolCallInformation(
            tools_called=False, tool_calls=[], content=model_output
        )
```

评论区精华

juliendenize 评论指出：最初实现用 getattr(request, "tool_choice", None) 兜底显得有些奇怪，既然用例是让 request 可以为 None，不如显式检查 request is None 并修改类型标注。作者 AndreasKaratzas 回复 "Updated the two Mistral leaf signatures :)"，将两个叶函数签名更新为显式 None 分支，随后获得 mgoin 与 sfeng33 的 APPROVED。

- _legacy_extract_tool_calls 中 request=None 的处理方式 (design): 作者采纳建议，将两个 Mistral 叶函数签名更新为显式 None 分支，并修改参数类型为 ChatCompletionRequest | None。

风险与影响

- 风险：
  1. 缺少直接配套单测：本 PR 没有新增测试文件，回归保护完全依赖 AMD CI Extended Generation 测试，后续 parser 重构可能再次破坏该兼容性。
  2. request 可空性放宽的影响面：_legacy_extract_tool_calls 现在接受 request=None 并返回 tools_called=False 的兜底结果，若调用方误传入 None 可能掩盖真实的参数错误，但该行为与旧实现一致，属于恢复而非新语义。
  3. bot_token_id 返回类型为 int | None：若外部调用方此前假定非空，统一重构后可能出现类型感知差异，但本 PR 只是恢复旧行为。
  4. 改动仅在 _is_pre_v11 分支生效，v11+ 路径走 super().extract_tool_calls_from_content，不受影响，整体回归风险可控。 - 影响：影响范围集中在 Mistral 工具调用解析路径：修复 AMD ROCm CI 的 Extended Generation 测试回归，恢复 Mistral pre-v11 tokenizer 模型在工具调用场景下的正常解析（包括 [TOOL_CALLS] 信封格式与 request 可选场景）。对使用 MistralToolParser 的 serving 层调用方是正向兼容性恢复；对 v11+ 解析路径无行为变化。团队层面消除了 AMD CI 的阻塞失败，属于小范围高价值修复。 - 风险标记：缺少直接单元测试配套，依赖 AMD CI 回归验证，parser API 可空性放宽

关联脉络

- PR #48947 Unified Mistral tool parser engine: PR body 通过 git bisect 定位的回归源头：统一 parser 将 bot_token / bot_token_id 移到嵌套 engine，并让 _legacy_extract_tool_calls 假定 request 对象存在；本 PR 正是为恢复这两项行为而修复。