PR #45003 完整报告

vllm-project/vllm

[Frontend] Support strict mode for tool calling

合并时间：2026-06-12 15:51

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45003

执行摘要

- 一句话：集成 xgrammar 结构标签实现工具调用严格模式
- 推荐动作：值得深度审查：设计上委托给 xgrammar 的思路清晰，代码简化效果显著，但默认启用严格模式的风险需要评估。建议在发布说明中明确提示，并考虑在下游版本中保留 VLLM_ENFORCE_STRICT_TOOL_CALLING 作为逃生门机制。此外，推理阶段约束的禁用原因值得团队进一步调研，以便未来恢复。

功能与动机

工具调用的严格模式可以强制模型输出符合 JSON Schema 的参数，提高可靠性和互操作性。之前 vLLM 依赖自定义的 StreamingXMLToolCallParser 进行流式 XML 解析，但维护成本高且与 xgrammar 功能重叠。通过集成 xgrammar 的内建结构标签，既能复用经过验证的约束引擎，又能覆盖更多模型（如 DeepSeek、Qwen、Llama 等），同时减少重复代码。PR body 中作者测试了 Multi-Minimax、Qwen 系列和 DeepSeek V3.2，均工作正常。cjackal 也确认 GLM-4.7/GLM-5 测试通过。

实现拆解

1. 删除废弃的自定义 XML 解析器：移除 vllm/tool_parsers/qwen3xml_tool_parser.py（1300 行）及对应测试文件，StreamingXMLToolCallParser 功能已由 Qwen3CoderToolParser 完全替代。
2. 构建结构标签注册中心：在 structural_tag_registry.py 中定义 XGRAMMAR_BUILTIN_STRUCTURAL_TAG_MODELS 和 VLLM_BUILTIN_STRUCTURAL_TAG_MODELS，提供 register_vllm_structural_tag 装饰器和 get_model_structural_tag 函数，优先查找 vLLM 注册表，未命中时委托给 xgrammar。
3. 扩展 ToolParser 基类：在 abstract_tool_parser.py 中新增 structural_tag_model 类属性和 get_structural_tag 方法，调整 adjust_request 流程以避免重复设置。
4. 在 DelegatingParser 中统一应用结构标签：在 abstract_parser.py 中新增 _apply_structural_tag 方法，在 adjust_request 内先于 tool_parser.adjust_request 执行，当前固定传入 reasoning=False 以规避生成循环。
5. 更新各模型工具解析器：包括 deepseekv4、qwen3coder、llama、hermes 等解析器，均设置正确的 structural_tag_model，清理旧的自定义 get_structural_tag 实现。

6. 调整测试：新增 tests/tool_parsers/test_structural_tag_registry.py 覆盖全部支持模型；修改 test_qwen3coder_tool_parser.py 移除对旧解析器的依赖；删除已废弃的 test_qwen3xml_tool_parser.py。

关键文件：

- vllm/tool_parsers/structural_tag_registry.py（模块 工具解析器；类别 source；类型 dependency-wiring；符号 register_vllm_structural_tag, get_model_structural_tag, XGRAMMAR_BUILTIN_STRUCTURAL_TAG_MODELS, VLLM_BUILTIN_STRUCTURAL_TAG_MODELS）：核心结构标签注册与分发入口，对接 xgrammar 内建模板和 vLLM 自有模板。
- vllm/tool_parsers/abstract_tool_parser.py（模块 工具解析器；类别 source；类型 core-logic；符号 structural_tag_model, get_structural_tag, init_subclass）：ToolParser 基类增加 structural_tag_model 和 get_structural_tag，定义子类接口。
- vllm/parser/abstract_parser.py（模块 解析层；类别 source；类型 core-logic；符号 _apply_structural_tag）：DelegatingParser 新增 _apply_structural_tag 方法，统一在 adjust_request 中应用结构标签。
- vllm/tool_parsers/qwen3xml_tool_parser.py（模块 工具解析器；类别 source；类型 deletion；符号 StreamingXMLToolCallParser, parse_single_streaming_chunks, _process_complete_xml_elements, _find_next_complete_element）：被删除的自定义 XML 解析器，约 1300 行代码被清除。
- tests/tool_parsers/test_structural_tag_registry.py（模块 工具解析器；类别 test；类型 test-coverage；符号 test_get_model_structural_tag_supports_all_xgrammar_builtins, test_get_model_structural_tag_supports_vllm_hermes, test_hermes_required_tool_calls_use_empty_separator, test_tool_parsers_declare_matching_xgrammar_builtin_model）：新增的测试文件，验证所有支持模型的结构标签生成正确。

关键符号：register_vllm_structural_tag, get_model_structural_tag, get_structural_tag, _apply_structural_tag, init_subclass

关键源码片段

vllm/tool_parsers/structural_tag_registry.py

核心结构标签注册与分发入口，对接 xgrammar 内建模板和 vLLM 自有模板。

```
def get_model_structural_tag(
    model: str,
    tools: list[ChatCompletionToolsParam] | None,
    tool_choice: ToolChoice,
    reasoning: bool,
) -> StructuralTag | None:
    """Build a structural tag with xgrammar's builtin model templates."""
    # 如果无 tools 或 tool_choice 是 "none"，直接返回 None
    if not tools or tool_choice == "none":
        return None
```

```
    # 将 vLLM 的 Pydantic 请求对象转换为 xgrammar 期望的 dict 格式
    dumped_tools = [_model_dump(tool) for tool in tools]
    dumped_tool_choice = _model_dump(tool_choice)

    if model in _VLLM_STRUCTURAL_TAG_REGISTRY:
        # 优先使用 vLLM 自有注册构建器（如 hermes）
        function_tools, builtin_tools, simplified_tool_choice = normalize_tool_choice(
            dumped_tools,
            dumped_tool_choice,
        )
        return _VLLM_STRUCTURAL_TAG_REGISTRY[model](
            function_tools,
            builtin_tools,
            simplified_tool_choice,
            reasoning,
        )

    # 检查模型是否在 xgrammar 内置列表中
    if model not in XGRAMMAR_BUILTIN_STRUCTURAL_TAG_MODELS:
        supported = sorted(SUPPORTED_STRUCTURAL_TAG_MODELS)
        raise ValueError(f"Unknown format type: {model}, supported types: {supported}")

    # 委托给 xgrammar 库内置的结构标签构建器
    return get_xgrammar_model_structural_tag(
        model=model,
        tools=dumped_tools,
        tool_choice=dumped_tool_choice,
        reasoning=reasoning,
    )
```

vllm/tool_parsers/abstract_tool_parser.py

ToolParser 基类增加 structural_tag_model 和 get_structural_tag，定义子类接口。

```
class ToolParser:
    # 新增类属性，子类通过设置此属性声明其匹配的 xgrammar 模板
    structural_tag_model: str | None = None

    def __init_subclass__(cls, **kwargs: Any) -> None:
        super().__init_subclass__(**kwargs)
        # 当启用严格模式时，自动禁用 supports_required_and_named，
        # 强制使用结构标签约束工具调用格式
        if (
            cls.structural_tag_model is not None
            and envs.VLLM_ENFORCE_STRICT_TOOL_CALLING
        ):
            cls.supports_required_and_named = False

    def get_structural_tag(
        self, request: ChatCompletionRequest, *, reasoning: bool = False
```

```
    ) -> StructuralTag | None:
        """根据模型的 structural_tag_model 从注册表获取 StructuralTag。"""
        if self.structural_tag_model is None:
            return None

        from vllm.tool_parsers.structural_tag_registry import (
            get_model_structural_tag,
        )

        return get_model_structural_tag(
            model=self.structural_tag_model,
            tools=request.tools,
            tool_choice=request.tool_choice,
            reasoning=reasoning,
        )
```

vllm/parser/abstract_parser.py

DelegatingParser 新增 _apply_structural_tag 方法，统一在 adjust_request 中应用结构标签。

```
  def _apply_structural_tag(
      self,
      request: ChatCompletionRequest | ResponsesRequest,
  ) -> ChatCompletionRequest | ResponsesRequest:
      """在 adjust_request 中早于 tool_parser.adjust_request 执行。
         只对 ChatCompletionRequest 且工具解析器声明了 structural_tag_model 时生效。"""
      if (
          not isinstance(request, ChatCompletionRequest)
          or self._tool_parser is None
          or self._tool_parser.structural_tag_model is None
          or not request.tools
      ):
          return request

      # 仅对需要工具调用的模式（auto/required/named）施加约束
      need_tool_calling = (
          request.tool_choice == "auto"
          or request.tool_choice == "required"
          or isinstance(request.tool_choice, ChatCompletionNamedToolChoiceParam)
      )
      if not need_tool_calling:
          return request

      # 获取结构标签，当前固定 reasoning=False 以避免生成循环
      structure_tag = self._tool_parser.get_structural_tag(
          request, reasoning=False,
      )
      if structure_tag is None:
          return request
```

```
    structural_tag = json.dumps(structure_tag.model_dump())
    request.structured_outputs = StructuredOutputsParams(
        structural_tag=structural_tag,
    )
    # 清除 response_format，避免与结构标签冲突
    request.response_format = None
    return request
```

评论区精华

- sfeng33 建议将结构标签设置逻辑从 ToolParser 移到 DelegatingParser，以便后续可以访问推理阶段的参数。作者采纳并实现了 _apply_structural_tag，但出于稳定性考虑，当前传递 reasoning=False。cjackal 也遇到类似问题并推测原因是结构标签屏蔽了非标准结束推理 token。
- cjackal 质疑移除 VLLM_ENFORCE_STRICT_TOOL_CALLING 环境变量会导致用户无法禁用严格模式，从而影响与 xgrammar 不兼容的配置（如流水线并行、推测解码）。作者回复“Done”，但最终仍移除了该变量，严格模式默认启用。
- yzong-rh 在 issue 评论中指出 xgrammar 的 structural_tag 与 vLLM 现有推理解析器（如 Minimax、Harmony）存在不兼容。例如 Minimax 的结构标签强制生成 \n

和

，但推理解析器认为推理只在 think 块内，导致标签强制生成的非推理部分被错误解析。作者承认问题并计划与 xgrammar 团队协商修复。

- 移除 VLLM_ENFORCE_STRICT_TOOL_CALLING 环境变量可能无法禁用严格模式 (design): 严格模式默认启用，但可能存在兼容性风险，后续可能需要提供开关。
- 结构标签应用位置选择 (design): 逻辑移至 abstract_parser.py 的 _apply_structural_tag，推理约束暂未启用。
- 推理阶段约束导致模型生成循环 (correctness): 当前禁用推理约束（reasoning=False），需进一步调查原因。

风险与影响

- 风险:
  1. 默认启用严格模式（移除了 VLLM_ENFORCE_STRICT_TOOL_CALLING）可能导致与 xgrammar 不兼容的配置（如流水线并行、推测解码）出现 500 错误，用户无法回退。
  2. 删除旧的 StreamingXMLToolCallParser 可能影响尚未迁移到 Qwen3CoderToolParser 的自定义解析器或外部集成。
  3. 推理阶段约束被禁用（reasoning=False）是因为已知生成循环问题，但长期会削弱约束力。
  4. xgrammar 内建结构标签与 vLLM 推理解析器之间的语义差异可能导致模型输出解析错误或 500 错误。
- 影响:

- 用户：工具调用行为变化，严格模式默认开启，可能提升生成参数格式的正确性，但也可能破坏现有灵活格式的工作流。用户可通过不提供 tools 或设置 tool_choice="none" 来绕过，但无法全局禁用。性能影响微小（基准测试显示 TTFT 和 TPOT 几乎一致）。
- 系统：代码库减少约 1300 行重复代码，工具调用格式约束统一由 xgrammar 处理，降低维护成本。新模型接入时只需注册结构标签构建器或选择 xgrammar 内置模板。
- 团队：需跟踪 xgrammar 更新，确保结构标签模板同步；需持续关注和修复推理解析器与结构标签之间的兼容性问题。
- 风险标记：默认启用严格模式无法关闭，删除旧解析器影响未迁移模型，推理阶段约束暂禁，xgrammar 标签与推理解析器不兼容

关联脉络

- PR #43678 Unknown (superseded by this PR): cjackal 在 issue 评论中指出本 PR 取代了 #43678