PR #40894 完整报告

vllm-project/vllm

feat: update xgrammar==0.2.0 to use structural tags for strict tool calling + reasoning for more models

合并时间：2026-05-05 03:45

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40894

执行摘要

- 一句话：升级 xgrammar 至 0.2.0，利用内置 structural tags 增强工具调用。
- 推荐动作：值得精读，尤其是 structural_tag_registry.py 中的装饰器注册模式和 adjust_request 的优先级设计。环境变量门控策略是降低风险的良好实践。建议关注后续的增量模型支持和 Qwen3Coder supports_required_and_named 的修复 PR。

功能与动机

PR 旨在利用 xgrammar 0.2.0 的最新特性 builtin_structural_tags，为更多模型的工具调用提供结构化生成。PR body 中提供了 benchmark 结果，例如 Qwen3.6-27B 的 Correct Call Rate 从 0.90 提升至 0.94，Correct Schema Rate 从 0.9375 提升至 1.0。

实现拆解

1. 升级依赖：在 requirements/common.txt 和 requirements/test/rocm.txt 中将 xgrammar 版本提升至 0.2.0，确保新 API 可用。
2. 新增注册框架：创建 vllm/tool_parsers/structural_tag_registry.py，实现装饰器 register_model_structural_tag 和查询函数 get_model_structural_tag，将模型名映射到标签构建函数。同时定义 _normalize_tool_choice 和 _get_function_parameters 等辅助函数，并内置 DeepSeek V4 的标签构建器 get_deepseek_v4_structural_tag（使用 xgrammar 的 TriggeredTagsFormat 等组件）。
3. 抽象基类扩展：在 vllm/tool_parsers/abstract_tool_parser.py 的 ToolParser 基类中添加虚方法 get_structural_tag（默认返回 None），并重写 adjust_request 方法：在 VLLM_ENFORCE_STRICT_TOOL_CALLING 为 True 且请求为 ChatCompletion 时，优先调用 get_structural_tag 获取标签并将其序列化后设置到 request.structured_outputs.structural_tag，避免覆盖用户已有的结构化输出配置。
4. 模型特定实现：在 qwen3coder_tool_parser.py 和 deepseekv4_tool_parser.py 中分别覆盖 get_structural_tag，委托到 registry 的 get_model_structural_tag 并传入对应的模型键（qwen_3_5 和 deepseek_v4）。同时将 Qwen3Coder 的 supports_required_and_named 设为 False，因为新增的结构化标签已覆盖该路径。
5. 环境变量与入口初始化：在 vllm/envs.py 中添加 VLLM_ENFORCE_STRICT_TOOL_CALLING 环境变量（默认 False）；在 vllm/entrypoints/openai/api_server.py 中调用 set_enable_structured_outputs_in_reasoning 从引擎配置同步推理内结构化输出开关。

6. 测试配套：在 tests/tool_parsers/test_qwen3coder_tool_parser.py 和 tests/tool_parsers/test_deepseekv4_tool_parser.py 中新增测试函数验证 structural tag 的正确生成、adjust_request 在 auto/required/named 工具选择下的行为，以及包含 reasoning 时的兼容性。

关键文件：

- vllm/tool_parsers/structural_tag_registry.py（模块 工具解析；类别 source；类型 new-module；符号 register_model_structural_tag, decorator, get_model_structural_tag, _normalize_tool_choice）：核心新增文件，定义了模型特定 structural tag 的注册和查询机制，是此 PR 的架构基础。
- vllm/tool_parsers/abstract_tool_parser.py（模块 工具解析；类别 source；类型 core-logic；符号 ToolParser.adjust_request, ToolParser.get_structural_tag）：核心逻辑变更：调整 adjust_request 方法加入 structural tag 路径，新增 get_structural_tag 虚方法，是工具解析流程的入口。
- vllm/tool_parsers/qwen3coder_tool_parser.py（模块 工具解析；类别 source；类型 core-logic；符号 Qwen3CoderToolParser.get_structural_tag, Qwen3CoderToolParser.supports_required_and_named）：为 Qwen3Coder 模型实现 get_structural_tag，连接 registry 并调整 supports_required_and_named 标志。
- vllm/tool_parsers/deepseekv4_tool_parser.py（模块 工具解析；类别 source；类型 core-logic；符号 DeepSeekV4ToolParser.get_structural_tag）：为 DeepSeek V4 模型实现 get_structural_tag，连接 registry 中的 deepseek_v4 构建器。
- vllm/envs.py（模块 环境变量；类别 source；类型 configuration；符号 VLLM_ENFORCE_STRICT_TOOL_CALLING）：新增环境变量 VLLM_ENFORCE_STRICT_TOOL_CALLING 控制全局切换。
- vllm/entrypoints/openai/api_server.py（模块 API 入口；类别 source；类型 entrypoint）：入口处根据引擎配置调用 set_enable_structured_outputs_in_reasoning 同步推理标志。
- tests/tool_parsers/test_qwen3coder_tool_parser.py（模块 测试套件；类别 test；类型 test-coverage；符号 test_get_vllm_registry_structural_tag_returns_structural_tag, test_adjust_request_auto_uses_vllm_registry_structural_tag, test_adjust_request_required_prefers_structural_tag）：为 Qwen3Coder 新增 structural tag 的单元测试，验证三种工具选择的标签生成和 adjust_request 集成。

关键符号：register_model_structural_tag, get_model_structural_tag, _normalize_tool_choice, set_enable_structured_outputs_in_reasoning, get_enable_structured_outputs_in_reasoning, get_deepseek_v4_structural_tag, ToolParser.get_structural_tag, ToolParser.adjust_request, Qwen3CoderToolParser.get_structural_tag, DeepSeekV4ToolParser.get_structural_tag

关键源码片段

vllm/tool_parsers/structural_tag_registry.py

核心新增文件，定义了模型特定 structural tag 的注册和查询机制，是此 PR 的架构基础。

```
# 文件：vllm/tool_parsers/structural_tag_registry.py
```

```
# 核心注册与查询机制

from collections.abc import Callable
from typing import Any, Literal

from xgrammar import StructuralTag
from xgrammar.structural_tag import (
    AnyTextFormat,
    ConstStringFormat,
    JSONSchemaFormat,
    SequenceFormat,
    TagFormat,
    TagsWithSeparatorFormat,
    TriggeredTagsFormat,
)
from vllm.entrypoints.openai.chat_completion.protocol import (
    ChatCompletionNamedToolChoiceParam,
    ChatCompletionToolsParam,
)

# 定义简化工具选择类型和构建器签名
SimplifiedToolChoice = Literal["auto", "required", "forced"]
ToolChoice = (
    Literal["none", "auto", "required"] | ChatCompletionNamedToolChoiceParam | None
)
StructuralTagBuilder = Callable[
    [list[ChatCompletionToolsParam], SimplifiedToolChoice, bool],
    StructuralTag,
]

# 全局注册表：模型名 -> 构建器
_structural_tag_registry: dict[str, StructuralTagBuilder] = {}


def register_model_structural_tag(name: str):
    """返回装饰器，将函数注册为模型 name 的 structural tag 构建器。"""
    def decorator(func: StructuralTagBuilder) -> StructuralTagBuilder:
        _structural_tag_registry[name] = func
        return func
    return decorator


def get_model_structural_tag(
    model: str,
    tools: list[ChatCompletionToolsParam] | None,
    tool_choice: ToolChoice,
    reasoning: bool,
) -> StructuralTag | None:
    """从注册表查找模型构建器，调用后返回 StructuralTag 对象；
```

```
    若 tools 为空或不需要工具调用则返回 None。"""
    builder = _structural_tag_registry.get(model)
    if builder is None:
        supported = list(_structural_tag_registry.keys())
        raise ValueError(f"Unknown format type: {model}, supported types: {supported}")

    normalized_tools, simplified_tool_choice = _normalize_tool_choice(
        tools=tools, tool_choice=tool_choice
    )
    if not normalized_tools:
        return None

    return builder(normalized_tools, simplified_tool_choice, reasoning)
```

vllm/tool_parsers/abstract_tool_parser.py

核心逻辑变更：调整 adjust_request 方法加入 structural tag 路径，新增 get_structural_tag 虚方法，是工具解析流程的入口。

```
# 文件：vllm/tool_parsers/abstract_tool_parser.py（部分）

from vllm.envs import VLLM_ENFORCE_STRICT_TOOL_CALLING
from vllm.sampling_params import StructuredOutputsParams

class ToolParser:
    # ... 其他代码

    def adjust_request(
        self,
        request: ChatCompletionRequest | ResponsesRequest,
    ) -> ChatCompletionRequest | ResponsesRequest:
        # 若无工具则直接返回
        if not request.tools:
            return request

        # Step 1（最高优先级）: 若启用严格工具调用且为 ChatCompletionRequest,
        # 尝试使用 vLLM 模型特定的 structural tag
        if (
            isinstance(request, ChatCompletionRequest)
            and VLLM_ENFORCE_STRICT_TOOL_CALLING
        ):
            need_tool_calling = (
                request.tool_choice == "auto"
                or request.tool_choice == "required"
                or isinstance(request.tool_choice, ChatCompletionNamedToolChoiceParam)
            )
            if need_tool_calling:
                structure_tag = self.get_structural_tag(request)
                if structure_tag is not None:
                    # 保留用户已有的 structured_outputs（如 regex/json），
```

```
                # 仅覆盖 structural_tag 字段
                if request.structured_outputs is None:
                    request.structured_outputs = StructuredOutputsParams(
                        structural_tag=json.dumps(structure_tag.model_dump()),
                    )
                else:
                    request.structured_outputs.structural_tag = json.dumps(
                        structure_tag.model_dump()
                    )
                return request

            # Step 2: 回退到基于工具 schema 的 JSON 指导解码
            json_schema_from_tool = get_json_schema_from_tools(
                tool_choice=request.tool_choice, tools=request.tools
            )
            # ... 后续标准逻辑
            return request

        def get_structural_tag(self, request: ChatCompletionRequest):
            """子类覆盖以返回模型特定的 StructuralTag，默认 None 表示不支持。"""
            return None
```

vllm/tool_parsers/qwen3coder_tool_parser.py

为 Qwen3Coder 模型实现 get_structural_tag，连接 registry 并调整 supports_required_and_named 标志。

```
# 文件：vllm/tool_parsers/qwen3coder_tool_parser.py（部分）

from vllm.tool_parsers.structural_tag_registry import (
    get_enable_structured_outputs_in_reasoning,
    get_model_structural_tag,
)

class Qwen3CoderToolParser(ToolParser):
    # 由于新增的 structural tag 已覆盖 required/named 工具选择，
    # 因此将原标志设为 False 以避免标准 JSON 回退路径干扰
    supports_required_and_named: bool = False

    def get_structural_tag(self, request: ChatCompletionRequest):
        """委托 registry 生成 Qwen3.5 格式的结构化标签。"""
        return get_model_structural_tag(
            model="qwen_3_5",
            tools=request.tools,
            tool_choice=request.tool_choice,
            reasoning=get_enable_structured_outputs_in_reasoning(),
        )
```

评论区精华

Review 中主要讨论了以下要点：

- 默认行为保护：aarnphm 要求将 VLLM_ENFORCE_STRICT_TOOL_CALLING 默认设为 False，避免影响现有用户。最终采纳。
- Qwen3Coder 的 supports_required_and_named：chaunceyjiang 指出将该标志设为 False 会破坏 tool_choice="required" 的现有行为，建议至少在 structural_tag 默认启用前保持 True，并引用 PR #42292 作为后续修正。该争议在 PR 内未完全解决，预计后续跟进。
- 非 xgrammar 后端兼容：sfeng33 询问如果使用非 xgrammar 后端是否会中断。设计上 structural tag 是 xgrammar 特性，但通过环境变量门控，未启用时完全走旧逻辑，不会强制使用。
- structured_outputs 覆盖：aarnphm 和 gemini-code-assist 指出直接覆盖 request.structured_outputs 会丢弃用户已有的 regex/json 设置。最终修改为仅更新 structural_tag 字段，保留其他配置。
- 错误信息修正：gemini 指出错误消息中引用了错误的方法名 get_xgrammar_builtin_structural_tag，已修正为 get_structural_tag。
  - VLLM_ENFORCE_STRICT_TOOL_CALLING 默认值 (design): 采纳，最终设为 False。
  - Qwen3Coder supports_required_and_named 设置 (design): 未在 PR 内修改，预计后续通过其他 PR 修复。
  - 非 xgrammar 后端兼容性 (design): 通过环境变量门控，未启用时完全回退旧逻辑，不会强制使用。
  - structured_outputs 覆盖问题 (design): 修改为仅更新 structural_tag 字段，保留其他配置。
  - 错误信息方法名错误 (style): 已修正为 get_structural_tag。

风险与影响

- 风险：
  1. 环境变量误配风险：若用户意外开启 VLLM_ENFORCE_STRICT_TOOL_CALLING，可能改变已有工具调用行为，导致输出格式与预期不符。
  2. Qwen3Coder 行为变更：supports_required_and_named = False 可能破坏依赖 tool_choice="required" 的工作流，需确保用户知晓或通过后续 PR 修复。
  3. xgrammar 版本兼容：升级到 0.2.0 可能引入新依赖问题；若用户环境中的 xgrammar 低于 0.1.34 则无法使用 get_model_structural_tag，但 PR 已提升最低版本要求。
  4. 测试覆盖不足：仅针对 Qwen3Coder 和 DeepSeek V4 新增了测试，未覆盖其他模型（如 GLM47）或非 xgrammar 后端场景。
  5. 推理兼容性：reasoning 模式下结构化输出的行为（set_enable_structured_outputs_in_reasoning）可能未充分验证，存在未知偏差。
- 影响：
  - 用户影响：默认无行为变化。开启新功能后，工具调用的格式正确率显著提升（参考 benchmark），但需承担 xgrammar 版本升级和配置调整的成本。

- 系统影响：新增约 330 行注册框架代码，对性能无直接影响。adjust_request 路径在工具调用时增加一次 registry 查询，开销可忽略。
- 团队影响：structural_tag_registry.py 为未来模型扩展提供了清晰的插件化接口，维护者需为每个支持的模型注册对应的标签构建器。讨论中提到的增量支持计划（如 GLM47）需后续跟进。
- 风险标记：环境变量门控行为改变风险，Qwen3Coder supports_required_and_named 回归，xgrammar 版本兼容性，测试覆盖仅限两个模型，推理兼容性未充分验证

关联脉络

- PR #42292 Fix Qwen3Coder supports_required_and_named setting: 讨论中 chaunceyjiang 引用此 PR，表示后续将修复 Qwen3Coder 的 supports_required_and_named 回归问题。