PR #25676 完整报告

sgl-project/sglang

Upgrade xgrammar to 0.2.1

合并时间：2026-05-29 11:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25676

执行摘要

- 一句话：升级 xgrammar 到 0.2.1，启用原生结构标签
- 推荐动作：值得精读。特别是 deepseekv32_detector.py 的删减方式和 kimik2_detector.py 中 get_structural_tag 的实现，展示了如何优雅地利用上游修复并处理兼容性问题。serving_chat.py 中的一行条件修改也体现了 reasoning 与 grammar 的职责分离设计。本 PR 是结构化标签功能演进的重要一步。

功能与动机

xgrammar 0.2.1 包含了重要的上游修复：mlc-ai/xgrammar#638 修正了 DeepSeek-V3.2 调用分隔符（从双换行改为单换行，避免贪婪解码下并行调用坍缩）；mlc-ai/xgrammar#623 为 Kimi auto 模式添加了 section markers 包装。升级后可以移除本地的 workaround 代码，直接利用 xgrammar 的内置结构标签，简化维护并提升兼容性。详见 PR 描述中的 Notes 部分。

实现拆解

1. 升级 xgrammar 版本：更新 python/pyproject.toml、python/pyproject_cpu.toml、python/pyproject_npu.toml、python/pyproject_other.toml、3rdparty/amd/wheel/sglang/pyproject.toml 以及 NPU CI 工作流（.github/workflows/nightly-test-npu.yml、full-test-npu.yml）中的 xgrammar pin 从 0.2.0 至 0.2.1。
2. 移除 DeepSeek-V3.2/V4 本地结构标签：在 deepseekv32_detector.py 中删除了长达 115 行的手工 get_structural_tag 实现（该实现曾用于规避 xgrammar 0.2.0 的双换行缺陷和缺失的 section wrapper）。添加了 get_structural_tag_name 方法返回 "deepseek_v3_2"（或 "deepseek_v4"），让 FunctionCallParser 直接调用 xgrammar 的 get_model_structural_tag 内置标签。
3. 为 Kimi K2/K2.5 添加原生结构标签：在 kimik2_detector.py 中新增 get_structural_tag 方法，当 tool_choice 为 "required" 或 ToolChoice 实例时，调用 get_model_structural_tag(model="kimi") 生成原生 Kimi 结构标签。对于非严格参数（strict=False），将参数 schema 替换为宽松的 {"type": "object"}，保证输出始终为 JSON 对象，兼容 Kimi 解析器。
4. 修正 reasoning parser 与 xgrammar 的协作：在 serving_chat.py 的 _process_messages 中，将 xgrammar_reasoning 的判断条件从 thinking_mode and (reasoning_parser is not None) 改为 thinking_mode and (reasoning_parser is None)。

此修正确保了当配置了自定义 reasoning parser（如 kimi_k2）时，xgrammar 不产生推理约束，推理前缀由 ReasonerGrammarBackend 控制。

5. 扩展测试覆盖：
   - 在 test_serving_chat.py 中添加了 7 个 Kimi 工具调用推理测试，验证默认 / 显式开启 / 关闭 thinking 参数的行为，并测试 xgrammar_tag_omits_reasoning_when_parser_owns_it 场景。
   - 在 test_function_call_parser.py 中调整了 DeepSeek 和 Kimi 的 test_get_model_structural_tag 测试，添加对结构标签序列化内容的验证（确认单换行分隔符）；并新增 test_kimi_required_no_strict_uses_empty_schema 等测试验证宽松对象 schema。

关键文件：

- python/sglang/srt/function_call/deepseekv32_detector.py（模块 函数调用检测；类别 source；类型 dependency-wiring；符号 get_structural_tag, _invoke_tag, get_structural_tag_name）：核心变更文件：移除了 115 行本地结构标签实现，切换为 xgrammar 内置标签，是升级的核心
- python/sglang/srt/function_call/kimik2_detector.py（模块 Kimi 检测器；类别 source；类型 core-logic；符号 get_structural_tag, get_structural_tag_name）：新增 get_structural_tag 方法，利用 xgrammar 原生 Kimi 结构标签，处理非严格参数兼容性
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 对话处理；类别 source；类型 core-logic；符号 _process_messages, _convert_to_internal_request）：修正 xgrammar_reasoning 判断条件，确保 reasoning parser 配置时 xgrammar 不注入推理
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 对话测试；类别 test；类型 test-coverage；符号 test_kimi_tool_call_keeps_default_reasoning, test_kimi_tool_call_keeps_explicit_reasoning, test_kimi_tool_call_respects_explicit_reasoning_disable, test_kimi_tool_call_keeps_template_default_thinking）：添加了 7 个 Kimi tool call 推理测试用例，覆盖默认 / 显式 reasoning 配置
- test/registered/unit/function_call/test_function_call_parser.py（模块 函数调用测试；类别 test；类型 test-coverage；符号 _constraint_json, test_kimi_routes_through_legacy_with_section_markers, test_kimi_routes_through_native_with_section_markers, test_kimi_required_no_strict_uses_empty_schema）：调整 DeepSeek 和 Kimi 的结构标签测试，添加序列化内容验证和宽松 schema 测试

关键符号：get_structural_tag, get_structural_tag_name, _process_messages, _convert_to_internal_request, test_kimi_tool_call_keeps_default_reasoning, test_kimi_tool_call_keeps_explicit_reasoning, test_kimi_tool_call_respects_explicit_reasoning_disable, test_kimi_routes_through_native_with_section_markers, test_kimi_required_no_strict_uses_loose_object_schema

关键源码片段

python/sglang/srt/function_call/kimik2_detector.py

新增 get_structural_tag 方法，利用 xgrammar 原生 Kimi 结构标签，处理非严格参数兼容性

```
# kimik2_detector.py (head 版本核心片段 )
# 使用 xgrammar 原生 Kimi 结构标签，并桥接非严格参数

def get_structural_tag(
    self,
    tools: Union[List[Tool], None] = None,
    tool_choice: Union[ToolChoice, Literal["auto", "required"]] = "auto",
    thinking_mode: bool = False,
) -> Optional[StructuralTag]:
    # 只有当 tool_choice 为 required 或 ToolChoice 时才使用原生标签;
    # auto 模式回退到基类 (legacy path，使用 structure_info 中的 section markers)
    if not (tools and (tool_choice == "required" or isinstance(tool_choice, ToolChoice))):
        return super().get_structural_tag(
            tools=tools, tool_choice=tool_choice, thinking_mode=thinking_mode
        )
    # 如果 xgrammar 未安装，返回 None
    if get_model_structural_tag is None:
        return None

    converted_tools = []
    for tool in tools:
        converted_tool = tool.model_dump()
        function = converted_tool["function"]
        if not function.get("strict", False):
            # Kimi 的解析器只接受对象形状的 tool 参数。XGrammar
            # 将 strict=False 的 argument 视为未约束 JSON，可能生成
            # 字符串 / 数组 / 数字，Kimi 无法解析。我们只约束外层类型
            # 为 object，保持非严格语义。
            function["strict"] = True
            function["parameters"] = _KIMI_NON_STRICT_ARGUMENTS_SCHEMA # {"type": "object"}
        converted_tools.append(converted_tool)

    converted_tool_choice = (
        tool_choice.model_dump() if isinstance(tool_choice, ToolChoice) else tool_choice
    )
    return get_model_structural_tag(
        model="kimi",
        tools=converted_tools,
        tool_choice=converted_tool_choice,
        reasoning=thinking_mode,
    )
```

python/sglang/srt/entrypoints/openai/serving_chat.py

修正 xgrammar_reasoning 判断条件，确保 reasoning parser 配置时 xgrammar 不注入推理

```
# serving_chat.py 中 _process_messages 方法的关键修正
```

```
# 当 reasoning parser 配置时，xgrammar 的 structural tag 只负责工具调用后缀，
# 推理前缀由 ReasonerGrammarBackend 负责，因此 xgrammar_reasoning 应该
# 在 reasoning_parser 为 None 时才为 True。
xgrammar_reasoning = thinking_mode and (
    self.tokenizer_manager.server_args.reasoning_parser is None
)
# 之前错误地使用了 is not None，导致自定义 reasoning parser 无法接管推理。
```

评论区精华

PR 的 Review 交流较少，两位 Reviewer 均给出了正面评价。Ubospica 表示 'This looks great to me!'。主要设计决策已在 PR body 中阐述：Kimi 非严格参数使用宽松对象 schema 作为兼容性桥接，以及 reasoning parser 优先控制推理前缀。作者在 issue 评论中提供了详细的验证结果（包括多轮 CI 运行和针对 Kimi 工具空参数的服务验证）。

- Kimi 非严格 tool choice 参数 schema 兼容性 (design): 采用宽松对象 schema 作为兼容性桥接，已在 empty arguments 场景验证。
- Reasoning parser 与 xgrammar 的职责划分 (design): 修正为 reasoning_parser is None 时才让 xgrammar 处理推理，已由测试覆盖。

风险与影响

- 风险：① 依赖升级风险：xgrammar 0.2.1 作为新依赖，虽包含必要修复，但仍可能存在未覆盖的模型组合的回归。已通过 PR Test (Extra) CI 和针对性服务验证降低风险。 ② Kimi 兼容性桥接：非严格参数使用 {"type": "object"} 作为宽松 schema，理论上可能在某些极端输入下输出不符合 Kimi 解析预期，但经过 empty arguments 场景验证。 ③ DeepSeek 行为变化：从本地 workaround 切换到 xgrammar 内置标签后，分隔符行为改变（双换行→单换行），已通过序列化断言验证。 ④ Reasoning 逻辑修正：serving_chat.py 中的一行条件改变影响了所有 reasoning parser 的交互，但已通过新测试覆盖。 ⑤ NPU/XPU 路径：XPU Docker 路径没有 xgrammar 安装，保持不变。NPU CI 中版本已更新，但未进行功能测试验证。
- 影响：用户影响：使用 DeepSeek-V3.2/V4 和 Kimi K2/K2.5 模型的用户将获得更可靠的工具调用，不再出现并行调用坍缩或无 section markers 的问题。 系统影响：减少了本地维护的代码量（-115 行），依赖上游修复，降低长期维护负担。 团队影响：需要关注 xgrammar 后续版本，因为本地 workaround 已移除，依赖上游更加紧密。 兼容性：对非 Kimi 或非 DeepSeek 模型的用户无影响。所有修改均与现有 API 接口兼容。
- 风险标记：依赖升级，兼容性桥接，Kimi 行为改变，核心路径变更

关联脉络

- 暂无明显关联 PR