PR #25100 完整报告

sgl-project/sglang

[model] Apertus Tool/Function and Reasoning parser

合并时间：2026-06-06 15:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25100

执行摘要

- 一句话：新增 Apertus2509 工具调用与推理块解析器
- 推荐动作：该 PR 设计思路清晰，遵循了现有解析器框架的扩展模式，是添加新模型格式的良好范例。建议阅读 apertus2509_detector.py 和 reasoning_parser.py 的源码，了解如何集成 tool call 与 reasoning 解析。对于需要支持相似自定义格式的开发者，该 PR 提供了可复用的模式。

功能与动机

Apertus 2509 模型在生成中使用自定义的特殊标记来分隔推理块（<|inner_prefix|>/<|inner_suffix|>）和工具调用（<|tools_prefix|>[/<|tools_suffix|>）。现有的通用格式检测器无法正确解析这些格式，导致推理内容和工具调用在输出中被混淆或丢失。该 PR 通过新增专门的检测器 / 解析器以及对服务管线的适配，使 Apertus 2509 模型能够与 SGLang 的 reasoning parser 和 function call parser 无缝集成。

实现拆解

1. 新增工具调用检测器：在 python/sglang/srt/function_call/apertus2509_detector.py 中实现 Apertus2509Detector，继承 BaseFormatDetector。支持一次性解析（detect_and_parse）和流式增量解析（parse_streaming_increment），识别 <|tools_prefix|>[{...}]<|tools_suffix|> 格式，并将每个 JSON 对象转换为 ToolCallItem。
2. 新增推理块解析器：在 python/sglang/srt/parser/reasoning_parser.py 中添加 Apertus2509Detector（继承 BaseReasoningFormatDetector）。实现了 detect_and_parse_block_sequence 方法，将文本拆分为有序的 ('reasoning', ...) 和 ('text', ...) 块，并支持流式增量解析（parse_streaming_increment）。同时提供 _split_inner_reasoning 方法，处理推理块内可能嵌入的工具调用。
3. 自动检测与路由：在 python/sglang/srt/managers/template_detection.py 中新增 _is_apertus2509 检测函数（基于词汇表是否包含 <|inner_prefix|>），并将 apertus2509 同时注册到 REASONING_PARSER_RULES 和 TOOL_CALL_PARSER_RULES，实现模型自动识别。
4. 服务路径适配：在 python/sglang/srt/entrypoints/openai/serving_chat.py 中将原有 _patch_mistral_skip_special_tokens 泛化为 _patch_reasoning_skip_special_tokens，对 apertus2509 解析器强制设置 skip_special_tokens=False，确保推理标记在解码时不被剥离。

5. API 扩展：在 python/sglang/srt/entrypoints/http_server.py 的 /separate_reasoning 端点增加 return_blocks 参数，当请求中包含该参数时，返回结构化的块列表（blocks、reasoning_blocks、text_blocks），使客户端能够保留交替的推理和工具调用片段。
6. 解析器注册与测试：在 python/sglang/srt/function_call/function_call_parser.py 中将 Apertus2509Detector 注册到 FunctionCallParser.ToolCallParserEnum；更新测试文件 test/registered/unit/managers/test_template_manager.py 以覆盖新检测规则。同时更新 docs_new 下的文档和 notebook 示例。

关键文件：

- python/sglang/srt/function_call/apertus2509_detector.py（模块 工具调用；类别 source；类型 core-logic；符号 Apertus2509Detector, init, has_tool_call, detect_and_parse）：核心新增文件，实现 Apertus 工具调用格式的检测与解析。
- python/sglang/srt/parser/reasoning_parser.py（模块 推理解析；类别 source；类型 core-logic；符号 Apertus2509Detector, init, _ends_with_partial_token, detect_and_parse）：核心修改文件，添加 Apertus 推理块解析器及块序列解析支持。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 服务入口；类别 source；类型 core-logic；符号 _patch_mistral_skip_special_tokens, _patch_reasoning_skip_special_tokens）：泛化 skip_special_tokens 补丁，确保 apertus2509 特殊标记不被剥离。
- python/sglang/srt/entrypoints/http_server.py（模块 服务入口；类别 source；类型 core-logic）：扩展 /separate_reasoning API 以支持 return_blocks 结构输出。
- python/sglang/srt/managers/template_detection.py（模块 模型识别；类别 source；类型 core-logic；符号 _is_apertus2509）：新增 apertus2509 自动检测规则（基于词汇表）。
- python/sglang/srt/function_call/function_call_parser.py（模块 工具调用；类别 source；类型 dependency-wiring）：将 Apertus2509Detector 注册到 ToolCallParserEnum。
- python/sglang/srt/managers/io_struct.py（模块 数据结构；类别 source；类型 core-logic）：添加 Apertus 相关配置（细节未在 patch 中详细展示）。
- test/registered/unit/managers/test_template_manager.py（模块 模板管理测试；类别 test；类型 test-coverage）：添加针对 apertus2509 检测规则的单元测试。
- docs_new/docs/advanced_features/separate_reasoning.mdx（模块 文档；类别 other；类型 documentation）：文档更新，添加 Apertus 2509 模型的使用说明。
- docs_new/docs/advanced_features/tool_parser.mdx（模块 文档；类别 other；类型 documentation）：文档更新，添加 Apertus 2509 工具解析说明。
- docs_new/docs/advanced_features/separate_reasoning.ipynb（模块 文档；类别 other；类型 documentation）：Notebook 示例更新。
- docs_new/docs/advanced_features/tool_parser.ipynb（模块 文档；类别 other；类型 documentation）：Notebook 示例更新。

关键符号：Apertus2509Detector.init, Apertus2509Detector.detect_and_parse, Apertus2509Detector.parse_streaming_increment, Apertus2509Detector.detect_and_parse_block_sequence, Apertus2509Detector._split_inner_reasoning, _patch_reasoning_skip_special_tokens,

_is_apertus2509

关键源码片段

python/sglang/srt/function_call/apertus2509_detector.py

核心新增文件，实现 Apertus 工具调用格式的检测与解析。

```
# python/sglang/srt/function_call/apertus2509_detector.py
class Apertus2509Detector(BaseFormatDetector):
    def __init__(self):
        super().__init__()
        self.bot = '<|tools_prefix|>['
        self.suffix = '<|tools_suffix|>'
        self._in_tools_block: bool = False

    def detect_and_parse(self, text: str, tools: List[Tool]) -> StreamingParseResult:
        '''一次性解析：提取所有工具块并解析 JSON 负载。'''
        if not self.has_tool_call(text):
            return StreamingParseResult(normal_text=text, calls=[])
        calls: List[ToolCallItem] = []
        normal_parts: List[str] = []
        cursor = 0
        while True:
            if (start := text.find(self.bot, cursor)) == -1:
                normal_parts.append(text[cursor:])
                break
            normal_parts.append(text[cursor:start])
            tool_part = text[start:]
            parsed_arr, json_end = self._try_parse_json_array(tool_part)
            if parsed_arr is None:
                normal_parts.append(tool_part)
                break
            if (suffix_pos := tool_part.find(self.suffix, json_end)) == -1:
                normal_parts.append(tool_part)
                break
            calls.extend(self._parse_apertus_call_list(parsed_arr, tools, tool_index_offset=len(calls))
            )
            cursor = start + suffix_pos + len(self.suffix)
        return StreamingParseResult(normal_text=''.join(normal_parts).strip(), calls=calls)

    # parse_streaming_increment 等略
```

python/sglang/srt/parser/reasoning_parser.py

核心修改文件，添加 Apertus 推理块解析器及块序列解析支持。

```
# python/sglang/srt/parser/reasoning_parser.py
class Apertus2509Detector(BaseReasoningFormatDetector):
    def __init__(self, stream_reasoning=True, force_reasoning=False,
                 continue_final_message=False, previous_content='',
```

```
            force_nonempty_content=False):
        super().__init__(
            '<|inner_prefix|>', # 推理块起始标记
            '<|inner_suffix|>', # 推理块结束标记
            force_reasoning=False,
            stream_reasoning=stream_reasoning,
            continue_final_message=continue_final_message,
            previous_content=previous_content,
        )
        self._force_reasoning = force_reasoning
        self._force_nonempty_content = force_nonempty_content
        self._tool_start_token = '<|tools_prefix|>['
        self._tool_end_token = '<|tools_suffix|>'
        self._reasoning_acc = ''
        self._in_inner_tool = False

    def detect_and_parse_block_sequence(self, text: str) -> list[tuple[str, str]]:
        '''返回有序块序列：[('reasoning'|'text', content), ...]'''
        start_tok = self.think_start_token
        end_tok = self.think_end_token
        blocks = []
        cursor = 0
        if self._in_reasoning: # 继续消息可能已在推理块内
            if (e := text.find(end_tok, cursor)) == -1:
                blocks.extend(self._split_inner_reasoning(text[cursor:]))
                blocks.append(('text', ''))
                return blocks
            blocks.extend(self._split_inner_reasoning(text[cursor:e]))
            cursor = e + len(end_tok)
        while True:
            if (s := text.find(start_tok, cursor)) == -1:
                blocks.append(('text', text[cursor:]))
                break
            if s > cursor:
                blocks.append(('text', text[cursor:s]))
            cursor = s + len(start_tok)
            if (e := text.find(end_tok, cursor)) == -1:
                blocks.extend(self._split_inner_reasoning(text[cursor:]))
                blocks.append(('text', ''))
                break
            blocks.extend(self._split_inner_reasoning(text[cursor:e]))
            cursor = e + len(end_tok)
        last_idx = len(blocks) - 1
        blocks = [(k, t) for i, (k, t) in enumerate(blocks)
                  if not (k == 'text' and t == '' and i != last_idx)]
        return blocks
```

评论区精华

Review 中，JustinTong0323 指出三个关键问题：

- skip_special_tokens 问题：普通聊天请求仍会解码时剥离特殊 token，导致推理解析器看不到标记，需要强制 skip_special_tokens=False。→ 已修复（_patch_reasoning_skip_special_tokens 中对 apertus2509 设置 False）。
- 文档路径问题：旧版 docs/ 下的更新需移至 docs_new/ 以避免 lint 拒绝。→ 已修复。
- 流式工具块后文本残留：parse_streaming_increment 中完成工具块后可能不及时刷新尾随普通文本。→ 已修复（'Fix review' 提交）。
- 推理检测器参数转发：Apertus2509Detector 构造函数需接收 continue_final_message 等 kwargs 并传递至基类。→ 已修复。 最终获两位 reviewer 批准合并。
- skip_special_tokens 导致推理标记被剥离 (correctness)：已修复：_patch_reasoning_skip_special_tokens 中对 apertus2509 解析器设置 skip_special_tokens=False。
- 文档路径需从 docs/ 迁移至 docs_new/ (documentation)：已修复：提交将文档更新移至 docs_new/。
- 流式解析中工具块后普通文本残留 (correctness)：已修复（在 'Fix review' 提交中）。
- 推理检测器构造函数需转发公共 kwargs (design)：已修复：__init__ 现在接受完整 kwargs 并传递至基类。

风险与影响

- 风险：
  1. 自动检测误触风险：_is_apertus2509 仅检查词汇表中是否存在 <linner_prefixl>，若其他模型意外包含该 token 则可能被错误识别，但概率极低。
  2. skip_special_tokens 影响：强制关闭特殊 token 剥离可能导致解码输出中出现其他特殊 token，但仅对 apertus2509 解析器生效，影响范围有限。
  3. 流式缓冲边界：parse_streaming_increment 中的缓冲管理逻辑若未正确处理 token 边界，可能导致工具调用解析不完整或推理块分裂。
  4. API 扩展兼容性：return_blocks 为新增参数，不影响现有客户端调用。 - 影响：对使用 Apertus 2509 模型的用户，该 PR 使工具调用和推理内容得到正确分离与流出；对系统，新增约 500 行代码，整体性能影响极小；对团队，需在未来维护该模型格式的解析逻辑，但代码结构与现有框架一致，学习成本低。 - 风险标记：特殊 token 剥离风险，自动检测误触发，流式边界 case

关联脉络

- 暂无明显关联 PR