PR #25876 完整报告

sgl-project/sglang

Fix Anthropic Messages API compatibility

合并时间：2026-06-13 05:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25876

执行摘要

- 一句话：修复并增强 Anthropic /v1/messages 兼容性
- 推荐动作：此 PR 价值极高，建议负责 API 兼容层的开发者精读。重点关注：
  - streaming 生成器中 content_block 状态管理设计（serving.py 的 _ensure_content_block_events）
  - 思考参数在 Anthropic 与 OpenAI 之间的映射机制（serving_chat.py 的 apply_reasoning_enabled）
  - 协议模型从普通 Pydantic 模型向 discriminated union 的迁移方式（protocol.py）
  - 测试用例的构造模式，尤其是如何用 fake serving 模拟底层行为。

功能与动机

Anthropic /v1/messages 端点存在多处兼容性问题：流式输出中 text_delta 可能误打到 tool_use 块索引；reasoning_content 未被映射为 thinking 块；缓存 token 重复计入 input_tokens；web_search 内置工具因缺少 input_schema 导致 422 验证错误；OpenAI handler 错误被吞掉返回通用 500。这些问题导致 Claude Code、anthropic-sdk-python 等客户端无法正常使用 SGLang 作为后端。此 PR 旨在系统性修复这些缺陷，使 Anthropic 兼容层达到可投产水平。

实现拆解

1. 协议模型重构（protocol.py）：将原有单一的 AnthropicContentBlock 拆分为 TextBlock、ToolUseBlock、ThinkingBlock 等带歧视联合的独立模型，采用 Pydantic Discriminator 实现类型安全解析。新增 ThinkingDelta、SignatureDelta 等流式事件类型。
2. 服务层核心逻辑重写（serving.py）：重写 _generate_anthropic_stream 流式生成器，通过 _ensure_content_block_events 管理块打开 / 关闭状态，确保 text_delta/thinking_delta/input_json_delta 路由到正确的块索引。新增 _anthropic_usage_from_openai 函数统一处理 token 用量转换（减去缓存 token），新增 _convert_assistant_thinking_blocks 将 prior-turn thinking 历史重包装为 标签。接入 continuous_usage_stats 以获取实时流式用量。
3. OpenAI 侧新接口（serving_chat.py）：新增 wrap_reasoning_history、_reasoning_default_mode、_get_reasoning_toggle_param、apply_reasoning_enabled 四个方法，使 Anthropic 层能通过统一接口控制模型的思考模式，并保证与

_get_reasoning_from_request 的读取逻辑一致。

4. HTTP 错误处理适配（http_server.py）：针对 /v1/messages 路径的 RequestValidationError 和 HTTPException，返回 Anthropic 格式的 {"type":"error","error":{...}} 响应，并利用 _scrub_error_message 清洗 5xx 内部错误细节。
5. 测试覆盖（test_serving.py）：新增 1300+ 行单元测试，涵盖工具流块切换、思考流事件序列、缓存用量、web_search 展平、错误转发、thinking 参数多回合历史等场景，所有测试通过 CI（base-a-test-cpu）。

关键文件：

- python/sglang/srt/entrypoints/anthropic/serving.py（模块 适配层；类别 source；类型 core-logic；符号 _cached_prompt_tokens, _anthropic_input_tokens, _anthropic_usage_from_openai, _scrub_error_message）：核心服务层，实现 Anthropic 请求到 OpenAI 的转换、流式响应生成、错误处理、token 用量计算等关键逻辑。改动量最大（+843/-208）。
- python/sglang/srt/entrypoints/anthropic/protocol.py（模块 协议模型；类别 source；类型 data-contract；符号 AnthropicContentBlock, TextBlock, ImageBlock, ToolUseBlock）：协议模型文件，将原本单一的 AnthropicContentBlock 拆分为 TextBlock、ToolUseBlock、ThinkingBlock 等 discriminated union 模型，新增 WebSearchTool、ThinkingDelta、SignatureDelta 等类型。是保障类型安全的基础。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 推理控制；类别 source；类型 core-logic；符号 wrap_reasoning_history, _reasoning_default_mode, _get_reasoning_toggle_param, apply_reasoning_enabled）：OpenAI 端适配接口，新增 wrap_reasoning_history、apply_reasoning_enabled 等方法，使 Anthropic 层能与底层聊天模板和推理检测器正确交互，是 thinking 参数映射的关键依赖。
- test/registered/unit/entrypoints/anthropic/test_serving.py（模块 测试；类别 test；类型 test-coverage；符号 _FakeOpenAIServingChat, _generate_chat_stream, apply_reasoning_enabled, wrap_reasoning_history）：全新的 1339 行单元测试，覆盖了工具流块切换、思考流事件序列、缓存用量、search_result 展平、错误转发、thinking 参数等多回合场景，使用 fake OpenAI 服务模拟底层行为。
- python/sglang/srt/entrypoints/http_server.py（模块 错误处理；类别 source；类型 core-logic；符号 _anthropic_validation_message, _anthropic_error_response）：修改通用 HTTP 错误处理器，对 /v1/messages 路径返回 Anthropic 格式的错误响应，并清洗 5xx 内部细节。新增 _anthropic_validation_message 和 _anthropic_error_response 辅助函数。

关键符号：_cached_prompt_tokens, _anthropic_input_tokens, _anthropic_usage_from_openai, _scrub_error_message, _text_from_search_result, _convert_assistant_thinking_blocks, _emit_user_message, _message_start_event, wrap_reasoning_history, _reasoning_default_mode, _get_reasoning_toggle_param, apply_reasoning_enabled, _anthropic_validation_message, _anthropic_error_response

关键源码片段

python/sglang/srt/entrypoints/anthropic/serving.py

核心服务层，实现 Anthropic 请求到 OpenAI 的转换、流式响应生成、错误处理、token 用量计算等关键逻辑。改动量最大（+843/-208）。

```
def _cached_prompt_tokens(usage) -> int:
    # 从 OpenAI usage 对象中提取缓存 token 数量
    prompt_tokens_details = getattr(usage, 'prompt_tokens_details', None)
    return getattr(prompt_tokens_details, 'cached_tokens', 0) or 0


def _anthropic_input_tokens(usage) -> int:
    # 计算 Anthropic 可见的 input_tokens：prompt_tokens 减去缓存命中部分，
    # 避免双倍计费。若缓存超过 prompt（上游 telemetry 异常），钳位为 0 并记录 warning。
    prompt = getattr(usage, 'prompt_tokens', 0) or 0
    cached = _cached_prompt_tokens(usage)
    if cached > prompt:
        logger.warning(
            'Cached tokens (%d) exceed prompt tokens (%d); clamping '
            'input_tokens to 0. This usually indicates an upstream '
            'telemetry bug.',
            cached,
            prompt,
        )
    return max(prompt - cached, 0)


def _anthropic_usage_from_openai(
    usage,
    *,
    include_input: bool,
    include_output: bool,
    force_zero_output: bool = False,
) -> AnthropicUsage:
    # 将 OpenAI usage 转换为 AnthropicUsage，可选包含 input/output。
    # 流式 message_delta 事件应省略 input_tokens（仅 output_tokens），
    # 非流式响应两者都包含。force_zero_output 用于 streaming 首条 chunk。
    if usage is None:
        return AnthropicUsage(
            input_tokens=0 if include_input else None,
            output_tokens=0 if include_output else None,
        )
    usage_fields: dict[str, int] = {}
    cached_tokens = _cached_prompt_tokens(usage)
    if include_input:
        usage_fields['input_tokens'] = _anthropic_input_tokens(usage)
        if cached_tokens:
            usage_fields['cache_read_input_tokens'] = cached_tokens
    if include_output:
```

```
            usage_fields['output_tokens'] = (
                0 if force_zero_output else (getattr(usage, 'completion_tokens', 0) or 0)
            )
        return AnthropicUsage(**usage_fields)
```

python/sglang/srt/entrypoints/anthropic/protocol.py

协议模型文件，将原本单一的 AnthropicContentBlock 拆分为 TextBlock、ToolUseBlock、ThinkingBlock 等 discriminated union 模型，新增 WebSearchTool、ThinkingDelta、SignatureDelta 等类型。是保障类型安全的基础。

```
# 使用 Pyadntic Discriminator 实现 discriminated union
# 每个变体只携带自身使用的字段，而非一个包含所有可选字段的大模型
from typing import Annotated, Literal, Optional, Union
from pydantic import BaseModel, Field, Tag

class TextBlock(BaseModel):
    type: Literal['text'] = 'text'
    text: str

class ToolUseBlock(BaseModel):
    type: Literal['tool_use'] = 'tool_use'
    id: str
    name: str
    input: dict[str, Any] = Field(default_factory=dict)

class ThinkingBlock(BaseModel):
    type: Literal['thinking'] = 'thinking'
    thinking: str
    signature: Optional[str] = None

class RedactedThinkingBlock(BaseModel):
    type: Literal['redacted_thinking'] = 'redacted_thinking'
    data: Optional[str] = None

AnthropicContentBlock = Annotated[
    Union[
        TextBlock, ImageBlock, ToolUseBlock, ToolResultBlock,
        ToolReferenceBlock, SearchResultBlock, ThinkingBlock,
        RedactedThinkingBlock,
    ],
    Field(discriminator='type'),
]
```

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

OpenAI 端适配接口，新增 wrap_reasoning_history、apply_reasoning_enabled 等方法，使 Anthropic 层能与底层聊天模板和推理检测器正确交互，是 thinking 参数映射的关键依赖。

```
def apply_reasoning_enabled(
    self, request: ChatCompletionRequest, enabled: bool
```

```
) -> None:
    """强制设置请求的推理开启/关闭状态，与 _get_reasoning_from_request 保持对称。"""
    if not self.reasoning_parser:
        if enabled:
            raise ValueError(
                'Anthropic thinking is not supported for models without '
                'a reasoning parser'
            )
        return

    # Hunyuan 用 reasoning_effort=medium/no_think
    if self.reasoning_parser == 'hunyuan':
        request.reasoning_effort = 'medium' if enabled else 'no_think'
        return

    # Mistral 用 reasoning_effort=medium/none
    config = self.template_manager.reasoning_config
    is_mistral = (config is not None and config.special_case == 'mistral') or (
        config is None and self._reasoning_default_mode() == 'mistral'
    )
    if is_mistral:
        request.reasoning_effort = 'medium' if enabled else 'none'
        return

    # 始终开启的模型无法关闭
    is_always_on = (config is not None and config.special_case == 'always') or (
        config is None and self._reasoning_default_mode() == 'always'
    )
    if is_always_on:
        if not enabled:
            raise ValueError(
                f'Reasoning parser "{self.reasoning_parser}" is always-on '
                f'and cannot be disabled via Anthropic thinking'
            )
        return

    # 通用 toggle_param 方式：设置 request.chat_template_kwargs
    toggle_param = self._get_reasoning_toggle_param()
    request.chat_template_kwargs = request.chat_template_kwargs or {}
    request.chat_template_kwargs[toggle_param] = enabled
```

评论区精华

在 Code Review 中，gemini-code-assist[bot] 指出基于 name.startswith('web_search') 检测内置工具可能过于宽泛，若用户自定义工具名以同样前缀开头会有冲突。作者 JustinTong0323 回应已采用 Pydantic 歧视联合体，通过

> 我们已从名称检测切换为基于 type 字段的 Pydantic 歧视联合体。类型模式 ^web_search_\d{8}$ 也在 Pydantic Field(pattern=...) 中强制执行，只有符合日期格式的

> Anthropic 类型才会匹配。同时，自定义工具即使 name 以 web_search 开头，只要其 type 不为空或为 'custom'，就不会被误判为内置工具。

该决策已被接受，未产生额外修改。

- 内置工具检测方式：基于 name 前缀 vs 基于 type 字段 (correctness): 作者已改为使用 Pydantic discriminated union，通过 type 字段的模式匹配（^web_search_\d{8}$）识别内置工具，同时自定义工具即使 name 以 web_search 开头，只要 type 不为空或为 'custom'，就不会被误判。

风险与影响

- 风险：
  1. 流式状态机复杂度（serving.py）：重写的流式生成器引入了 _ensure_content_block_events 状态管理，若边缘情况（如连续 tool_use 块、thinking 块与文本块交错）处理不当，可能导致 SSE 事件顺序违反 Anthropic 协议，进而导致客户端解析失败。测试已覆盖主要的块切换场景，但未覆盖所有异常顺序组合。
  2. token 计量准确性与潜在回归（serving.py）：_anthropic_input_tokens 从 prompt_tokens 中减去 cached_tokens 以避免双计，但若上游缓存的 token 数超过 prompt_tokens（如 telemetry bug），会钳位为 0 并记录 warning。这可能在缓存命中率高时输出 token 数为 0，影响客户端计费逻辑。
  3. thinking 参数与 OpenAI 推理控制的同步（serving_chat.py）：apply_reasoning_enabled 需要与 _get_reasoning_from_request 保持逻辑一致，若 future 修改其中一方而未同步另一方，可能造成 thinking 控制失效。作者已在 docstring 中强调同步要求，但缺乏自动化防护。
  4. 错误类型映射不完整（http_server.py）：_anthropic_error_response 对部分 HTTP 状态码（如 413 Request Too Large）未映射，会 fallback 到 api_error，可能导致客户端无法准确区分错误类型。 - 影响：对用户：原本无法正常使用的 Anthropic API 端点现在与官方 SDK 和 Claude Code 兼容，支持流式思考块、缓存 token 报告、web_search 工具等关键功能。用户无需修改客户端代码即可获得更完整的 Anthropic 体验，错误消息也更具可读性。对系统：改动集中在入口适配层，不涉及推理引擎核心，对性能影响可忽略。新引入的 continuous_usage_stats 会增加一点流式 token 计算开销，但已在 OpenAI 侧启用，此处只是同步启用。对团队：合并了大量来自社区的 PR 碎片，降低了后续维护多个分支的成本。但新的流式状态机和 discriminated union 模型增加了代码理解难度。
- 风险标记：流式状态机复杂度，token 计量异常钳位，思考控制同步风险，错误状态码映射不完整

关联脉络

- PR #24294 fix(anthropic): track block type and surface reasoning_content as thinking (#24293): 被整合的 PR：流式内容块类型追踪和 reasoning_content 映射为 thinking 块。

- PR #20830 fix: populate input_tokens in Anthropic streaming message_start: 被整合的 PR：在 message_start 中提供真实的 input_tokens。
- PR #20856 fix: populate cache_read_input_tokens in Anthropic /v1/messages responses: 被整合的 PR：将缓存 token 映射为 cache_read_input_tokens。
- PR #22030 [fix] Anthropic /v1/messages endpoint: forward original error instead of returning generic 500: 被整合的 PR：转发原始错误而非返回通用 500。
- PR #23024 fix: support Anthropic web_search built-in tools in /v1/messages: 被整合的 PR：支持 web_search 内置工具和 search_result 块。
- PR #19334 [Anthropic] Support thinking/reasoning tokens in /v1/messages: 被整合的 PR：thinking 参数支持的基础提案。
- PR #21902 feat(anthropic): add thinking/reasoning support and cache token usage to /v1/messages: 被整合的 PR：综合 thinking + cache token 用量，多轮思考历史重构模式。
- PR #22061 [Anthropic] Add thinking support for /v1/messages: 被整合的 PR：AnthropicThinkingParam、apply_reasoning_enabled 分解、count_tokens thinking 传播。
- PR #22135 [Anthropic] Support thinking/reasoning tokens in /v1/messages: 被整合的 PR：独立的 thinking/reasoning 支持方案，为合并设计提供了信息。
- PR #22375 fix(anthropic): use monotonic_time in Anthropic API entrypoint for accurate Prometheus metrics: 被整合的 PR：清理已死的 received_time_perf/validation_time 计时重复代码。
- PR #25271 feat(anthropic): add Claude 4.7 adaptive thinking support: 后续 PR：在基本 thinking 支持之上扩展 adaptive thinking 等特性（未包含在本 PR 中）。