PR #29579 完整报告

sgl-project/sglang

[Feature] Add --default-chat-template-kwargs server arg

合并时间：2026-07-14 02:34

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29579

执行摘要

- 一句话：新增 --default-chat-template-kwargs 服务级默认参数
- 推荐动作：值得精读。变更虽小（3 个文件、78 行），但完整展示了一个服务级默认配置的落地过程：单一共享入口注入 + setdefault 保证 per-request 优先级 + 对 " 字段被提前 pop" 的时序陷阱做显式镜像。关注点包括：(a) _process_messages 作为唯一注入点的覆盖面；(b) reasoning_effort 被 _convert_to_internal_request 提前弹出的处理方式；(c) 配置校验放在 ServerArgs.__post_init__ 还是使用方 __init__ 的风格分歧（merrymercy 反对仍被合并）；(d) 测试只覆盖函数层，未覆盖 CLI 解析。对需要实现 " 服务级默认参数 " 的开发者是很好的参照。

功能与动机

PR body 指出：目前没有途径在服务启动时一次性禁用思考（或设置任意 chat template kwarg）并作用于每个请求。运维只能 (a) 在每个客户端请求上传 chat_template_kwargs={"enable_thinking": false}，或 (b) 设置 SGLANG_DEFAULT_THINKING 环境变量——但该变量只覆盖 thinking 键（Qwen3/GLM-4.5/GLM-5.2 使用的是 enable_thinking），且不是 CLI flag。因此需要一个 --default-chat-template-kwargs 启动参数，例如 --default-chat-template-kwargs '{"enable_thinking": false}'，让默认值应用到所有请求。PR 同时明确了优先级：per-request chat_template_kwargs > reasoning_effort > --default-chat-template-kwargs > 模板默认。

实现拆解

1. 参数定义（python/sglang/srt/server_args.py）：在 ServerArgs.reasoning_parser 之后新增 default_chat_template_kwargs 字段，类型 Optional[Dict[str, Any]]，通过 Arg(type_parser=json.loads) 接收 CLI JSON 参数；帮助文本说明 " 应用于所有未被逐请求覆盖的请求、键需匹配模型 chat template、per-request 优先 "，默认值为 None。
2. 启动期校验（server_args.py 的 post_init）：在 _handle_asr_validation() 之后插入 isinstance(..., dict) 检查，非 dict 时抛出 ValueError（--default-chat-template-kwargs must decode to a JSON object）。原因是 json.loads 可解析出字符串、数组、布尔等非对象值，不拦截的话会在第一个请求执行 .items() 时抛 AttributeError，而 false/[] 还会被 or {} 静默当作空默认；启动期 fail-fast 让错误第一时间暴露。
3. 服务端读取（serving_chat.py 的 init）：OpenAIServingChat.__init__从 tokenizer_manager.server_args.default_chat_template_kwargs 读取，并用 or {} 归一化

为 dict 存为 self.default_chat_template_kwargs，后续合并逻辑无需再判空。

4. 请求注入（serving_chat.py 的 _process_messages 开头）：_process_messages 是 OpenAI Chat/Responses/Anthropic/tokenize 路径的共享入口，在函数顶部把默认值 setdefault 进 request.chat_template_kwargs（先 dict(request.chat_template_kwargs or {}) 拷贝，避免原地修改原对象），逐请求键天然优先；随后如果合并结果含 reasoning_effort 且 request.reasoning_effort 为 None，则镜像到 request.reasoning_effort。原因在于 _convert_to_internal_request 会在 _process_messages 之前从 chat_template_kwargs 弹出 reasoning_effort，默认值若不显式镜像将无法被 _get_reasoning_from_request 感知。
5. 测试配套（test/registered/unit/entrypoints/openai/test_serving_chat.py）：_MockTokenizerManager.server_args 增加 default_chat_template_kwargs=None；新增 3 个测试分别断言：请求未传时默认 enable_thinking=False 透传到 apply_chat_template；请求显式传 True 时覆盖默认；默认 reasoning_effort 被镜像到 request.reasoning_effort。测试通过 mock apply_chat_template 的 call_args.kwargs 验证，无需真实模型。

关键文件：

- python/sglang/srt/server_args.py（模块 配置解析；类别 source；类型 configuration；符号 ServerArgs.default_chat_template_kwargs, ServerArgs.post_init）：新增 --default-chat-template-kwargs 参数定义（type_parser=json.loads）并在 __post_init__ 中加入 JSON 对象类型校验，是功能的配置入口与 fail-fast 保障。
- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 对话服务；类别 source；类型 core-logic；符号 OpenAIServingChat.init, OpenAIServingChat._process_messages）：在 _process_messages 共享入口以 setdefault 语义合并默认 chat template kwargs，并显式镜像默认 reasoning_effort，是功能的核心注入逻辑。
- 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_default_chat_template_kwargs_applied_when_request_unset, test_default_chat_template_kwargs_overridden_per_request, test_default_chat_template_kwargs_mirrors_reasoning_effort）：3 个新单元测试覆盖默认应用、逐请求覆盖与 reasoning_effort 镜像，是行为契约的验证。

关键符号：ServerArgs.post_init, OpenAIServingChat.init, OpenAIServingChat._process_messages, test_default_chat_template_kwargs_applied_when_request_unset, test_default_chat_template_kwargs_overridden_per_request, test_default_chat_template_kwargs_mirrors_reasoning_effort

关键源码片段

python/sglang/srt/server_args.py

新增 --default-chat-template-kwargs 参数定义（type_parser=json.loads）并在 __post_init__中加入 JSON 对象类型校验，是功能的配置入口与 fail-fast 保障。

```
# python/sglang/srt/server_args.py
```

```
class ServerArgs:
    # 服务级默认 chat template kwargs：启动时传入一次，作用于所有请求。
    # 例如 --default-chat-template-kwargs '{"enable_thinking": false}'。
    # 键名必须匹配具体模型的 chat template（enable_thinking / thinking /
    # reasoning_effort 等），逐请求的 chat_template_kwargs 优先。
    default_chat_template_kwargs: A[
        Optional[Dict[str, Any]],
        Arg(
            help="Default chat template kwargs applied to every request when not "
            "overridden per-request. Keys must match what the model's chat template "
            "expects (e.g. enable_thinking, thinking, reasoning_effort). Per-request "
            "chat_template_kwargs takes precedence.",
            type_parser=json.loads,
        ),
    ] = None

    def __post_init__(self):
        ...
        self._handle_asr_validation()

        # json.loads 可以解析出任意 JSON 值（字符串、数组、布尔等），
        # 但这里只接受 JSON object。提前校验能让配置错误在启动阶段暴露，
        # 而不是等第一个请求到达 _process_messages 时执行 .items() 才崩溃；
        # 同时避免 false / [] 被 `or {}` 静默当作空默认。
        if self.default_chat_template_kwargs is not None and not isinstance(
            self.default_chat_template_kwargs, dict
        ):
            raise ValueError(
                "--default-chat-template-kwargs must decode to a JSON object"
            )
        ...
```

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

在 _process_messages 共享入口以 setdefault 语义合并默认 chat template kwargs，并显式镜像默认 reasoning_effort，是功能的核心注入逻辑。

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

class OpenAIServingChat(OpenAIServingBase):
    def __init__(self, tokenizer_manager, template_manager):
        super().__init__(tokenizer_manager)
        self.template_manager = template_manager
        self.tool_call_parser = self.tokenizer_manager.server_args.tool_call_parser
        self.reasoning_parser = self.tokenizer_manager.server_args.reasoning_parser
        # 没有配置时归一化为 {}，后续合并逻辑无需再判空。
        self.default_chat_template_kwargs = (
            self.tokenizer_manager.server_args.default_chat_template_kwargs or {}
        )
        ...
```

```
def _process_messages(self, request, is_multimodal):
    """Process chat messages and apply chat template"""
    # 服务级默认值注入：_process_messages 同时服务 OpenAI Chat /
    # Responses / Anthropic / tokenize 等路径，是唯一的注入点。
    # 使用 setdefault 语义，保证逐请求的 chat_template_kwargs 优先。
    if self.default_chat_template_kwargs:
        ctk = dict(request.chat_template_kwargs or {})
        for k, v in self.default_chat_template_kwargs.items():
            ctk.setdefault(k, v)
        request.chat_template_kwargs = ctk
        # _convert_to_internal_request 会在 _process_messages 之前从
        # chat_template_kwargs 弹出 reasoning_effort，因此默认值必须显式
        # 镜像到 request.reasoning_effort，下游 _get_reasoning_from_request
        # 才能感知（例如 dsv4 / Hunyuan / Mistral 的思考强度）。
        effort = ctk.get("reasoning_effort")
        if effort is not None and request.reasoning_effort is None:
            request.reasoning_effort = effort
    ...
```

评论区精华

核心讨论集中在四个点：

- reasoning_effort 时序问题（correctness）：gemini-code-assist[bot] 指出 _convert_to_internal_request 在 _process_messages 之前就从 chat_template_kwargs 弹出 reasoning_effort，因此默认值留在 chat_template_kwargs 里不会被提取，Hunyuan/Mistral 等模型会表现错误。作者回复已在 c69f5815 修复：合并后显式镜像到 request.reasoning_effort，并新增测试。
- JSON 类型校验（correctness）：gemini-code-assist[bot] 与 issue 评论者 ronhuafeng 都指出 json.loads 不保证是 JSON object，例如 --default-chat-template-kwargs '"foo"' 会解析成功后在 .items() 崩溃，false/[] 则被 or {} 静默吞掉。最终实现把校验放在 ServerArgs.post_init，与作者最初回应的 " 在 OpenAIServingChat.__init__抛 ValueError" 位置不同。
- 键名模型差异（design）：alexnails 质疑 --default-chat-template-kwargs '{"reasoning_effort": "none"}' 对 Qwen3/GLM 家族不会禁用 thinking（它们读 enable_thinking），并问 reasoning_effort 是否要传给 apply_chat_template。作者确认机制通用、reasoning_effort 经 _apply_jinja_template 的 extra_template_kwargs 传入模板，并修改帮助文本避免绑定模型族。
- 覆盖范围与风格（question/style）：alexnails 问是否覆盖 ollama /api/chat 与 embedding 入口，作者表示不在本 PR；merrymercy 反对在 server_args.py 里直接放 " random asserts"，认为破坏风格指南，最终 head 仍保留该校验且 PR 已合并。
  - 默认 reasoning_effort 在 _process_messages 前被弹出，无法到达下游 (correctness): 已解决：合并后显式镜像 reasoning_effort 到 request.reasoning_effort。

- json.loads 不保证 JSON 对象，非 dict 输入导致运行期崩溃 (correctness): 已解决：在 ServerArgs.__post_init__中增加 isinstance 检查，非 dict 时抛出 ValueError（ --default-chat-template-kwargs must decode to a JSON object）。
- 默认 reasoning_effort 对 Qwen3 / GLM 家族无效（enable_thinking vs reasoning_effort 键差异）(design): 已澄清并修改帮助文本（b87e1de0），改为通用表述，不绑定具体模型族。
- 是否覆盖 ollama /api/chat 与 embedding 入口 (question): 未处理，作为后续扩展方向；目前默认值只作用于 OpenAI 兼容 chat 入口。
- server_args.py 中直接插入校验语句的风格争议 (style): 最后合并的 head 版本中该校验仍保留在 __post_init__主流程内，PR 仍被合入；风格问题悬而未决。

风险与影响

- 风险:
  - 共享入口热路径变更：_process_messages 是所有 OpenAI 兼容 chat 请求的共享方法，合并逻辑在这里对每个请求执行一次 dict 操作，虽无性能风险，但 request.chat_template_kwargs 会被原地改写（setdefault 只增键），依赖同一 request 对象的后续代码（如日志、重试）会看到默认值被注入。
  - 默认值全局生效影响 reasoning：设置 {"enable_thinking": false} 会静默关闭所有请求的思考；若运维误以为 reasoning_effort="none" 对 Qwen3/GLM 家族有效，会得到不符合预期的行为（讨论中已澄清但代码层未做防错）。键名与模型强相关，配置错误只会在推理结果层面暴露。
  - 校验覆盖不全：类型校验放在 ServerArgs.post_init，绕过 ServerArgs 直接构造 OpenAIServingChat 的场景下 or {} 对字符串等 truthy 非 dict 值仍不拦截，可能继续触发 AttributeError；当前生产路径都经 ServerArgs，风险较低。
  - 测试缺口：测试只覆盖 serving_chat 函数层，缺少 CLI 参数解析的端到端用例（如非 dict 输入的报错信息、reasoning_effort 与 enable_thinking 键差异）；作者在 review 中提到的 test_default_chat_template_kwargs_init_rejects_non_dict 未出现在最终 head 中。
- 影响:
  - 用户 / 运维：新增一个启动参数即可统一所有 OpenAI 兼容 chat 请求的 chat template 行为，对推理模型禁用 thinking 的部署是直接收益；优先级语义清晰（per-request > reasoning_effort > 默认值 > 模板默认）。
  - 系统：OpenAI 兼容服务层行为可被默认参数整体改变；配置非法时服务器启动失败（fail-fast），比运行期崩溃更友好。
  - 团队：提供了 " 服务级默认注入 " 的参考模式；alexnails 提出的 ollama /api/chat 与 embedding 入口覆盖问题，是后续把逻辑抽成共享 util 的潜在演进方向。
  - 影响范围：中等，仅涉及 OpenAI 兼容 chat 入口与启动配置，不涉及调度、显存、kernel 等核心路径。
  - 风险标记：共享入口热路径变更，默认值全局影响 reasoning 行为，配置校验存在风格争议，缺少 CLI 级端到端测试，ollama 与 embedding 入口未覆盖

关联脉络

- PR #33351 [misc] Deep-merge nested config overrides and parse request bodies with orjson: 同一时期服务端配置解析与覆盖合并主题，且都涉及 HTTP 服务端 JSON 解析与默认 / 覆盖语义。
- PR #33334 config: stop writing config onto the published ServerArgs at three sites: 围绕 ServerArgs 配置写入与校验治理，本 PR 新增的 default_chat_template_kwargs 正是 ServerArgs 新增字段，属于同一配置治理演进线。
- PR #33336 config: keep runtime hicache and weight-version updates off ServerArgs: ServerArgs 配置作用域与更新路径的长期治理，与新增 ServerArgs 字段的校验 / 发布约定相关。