PR #33146 完整报告

sgl-project/sglang

Support thinking budget for Inkling

合并时间：2026-08-11 02:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33146

执行摘要

- 一句话：修复 thinking budget 静默失效，新增 Inkling 支持
- 推荐动作：值得精读。该 PR 展示了一条关于特殊 token 语义与拼接序列状态判定的典型教训：用 in 判定块开关状态在 token 复用场景下会静默失效，_open_thinking_start 的反向扫描是简洁的修复方案。值得关注的设计决策包括：将状态判定收敛为独立辅助函数、对单 delimiter 序列保持行为不变的兼容性承诺、以及用测试将子类 token ID 与 tokenizer 常量绑定。

功能与动机

PR body 明确说明："Inkling has no thinking-budget processor. Adding one surfaced a base-class bug: a thinking block counts as closed if the end token appears anywhere in prompt + output, and Inkling's end token also terminates every message, so thinking_budget was silently ignored." 即 Inkling 缺少 thinking budget 处理器，而为其添加时发现基类判定逻辑有缺陷——cur_ids 是 prompt 与 output 的拼接，只要任意位置出现 end token 就认为 thinking 块已关闭。对 GLM-4.5 / Qwen3 / DeepSeek-R1 这类 end token 只在 thinking 块尾部出现的模型没有问题，但对消息级终止符模型（Inkling 每条历史消息都以 end token 结尾）整个预算机制被永久旁路。

实现拆解

1. 定位根因（python/sglang/srt/sampling/custom_logit_processor.py）：ThinkingBudgetLogitProcessor.__call__ 原来的状态判定是 THINKING_START_TOKEN_ID not in cur_ids or THINKING_END_TOKEN_ID in cur_ids，随后用 cur_ids.index(THINKING_START_TOKEN_ID) 取第一个 start 作为计数起点。这隐含假设 end token 只出现在 thinking 块的闭合处，对消息级终止符模型不成立。
2. 新增 _open_thinking_start：从 ids 末尾反向扫描，先遇到 start 返回其索引（块未闭合），先遇到 end 返回 -1（块已闭合）。基类 __call__ 改用该函数判定状态并取计数起点，语义变为 " 从最近一个未闭合的 thinking start 开始计数 "；对只有一个 delimiter 的序列（GLM-4.5 / Qwen3 / DeepSeek-R1 的正常形态）结果与原来完全一致。
3. 新增 InklingThinkingBudgetLogitProcessor 子类：仅声明三个 token ID（start = 200008、end = 200010、newline = 198），复用基类预算控制逻辑，无额外逻辑。
4. 测试配套（test/registered/unit/sampling/test_custom_logit_processor.py）：原 test_multiple_thinking_start_counts_from_first 在两种语义下都通过，被重命名并重新设

计预算参数为 test_multiple_thinking_start_counts_from_most_recent，以锁定 " 从最近 start 计数 " 的新语义；新增 test_inkling_end_token_in_prompt_does_not_disable_budget，直接实例化 Inkling 子类、将 token ID 与 sglang.srt.parser.inkling_tokenizer 导出的 INKLING_SPECIAL_TOKEN_IDS 断言一致，并依次走通强制 newline 与强制 end 两条分支。测试文件相应新增了从 inkling_tokenizer 的导入。

5. 验证：PR body 给出 2×B200 上 Inkling-Small-NVFP4、reasoning_effort: high、thinking_budget: 60 的对照实验——不设置预算 8/8 超预算，main 上的旧逻辑 8/8 超预算，本 PR 0/8 超预算；24 次回答全部正确。单元测试 35 个全部通过。

关键文件：

- python/sglang/srt/sampling/custom_logit_processor.py（模块 采样器；类别 source；类型 core-logic；符号 _open_thinking_start, InklingThinkingBudgetLogitProcessor, ThinkingBudgetLogitProcessor.call）：核心源码：新增 _open_thinking_start 反向扫描函数修复基类 delimiter 判定 bug，并新增 InklingThinkingBudgetLogitProcessor 子类。这是整个 PR 的实质变更所在，影响所有 ThinkingBudgetLogitProcessor 子类的行为语义。
- test/registered/unit/sampling/test_custom_logit_processor.py（模块 采样测试；类别 test；类型 test-coverage；符号 test_multiple_thinking_start_counts_from_first, test_multiple_thinking_start_counts_from_most_recent, test_inkling_end_token_in_prompt_does_not_disable_budget）：测试配套：重命名并重新设计 test_multiple_thinking_start_counts_from_first 为 test_multiple_thinking_start_counts_from_most_recent 以锁定新语义，新增 test_inkling_end_token_in_prompt_does_not_disable_budget 直接覆盖 Inkling 子类、断言 token ID 与 tokenizer 常量一致、走通强制 newline 与强制 end 两条分支。

关键符号：_open_thinking_start, ThinkingBudgetLogitProcessor.call, InklingThinkingBudgetLogitProcessor

关键源码片段

python/sglang/srt/sampling/custom_logit_processor.py

核心源码：新增 _open_thinking_start 反向扫描函数修复基类 delimiter 判定 bug，并新增 InklingThinkingBudgetLogitProcessor 子类。这是整个 PR 的实质变更所在，影响所有 ThinkingBudgetLogitProcessor 子类的行为语义。

```
def _open_thinking_start(ids: list[int], start_id: int, end_id: int) -> int:
    """定位当前仍处于打开状态的 thinking 块的起始 token 索引；若没有则返回 -1。

    从序列末尾向前扫描：最后一个出现的分隔符决定状态——
    若最后出现的是 start_id，说明 thinking 块尚未闭合，返回其索引；
    若最后出现的是 end_id，说明最近一个 thinking 块已结束，返回 -1。
    这样即使 prompt 中自带 end token（如 Inkling 的 end token 兼作消息
    终止符，每条历史消息后都会出现），也不会误判当前块已关闭。
    """
    for idx in reversed(range(len(ids))):
        if ids[idx] == start_id:
            return idx
```

```
        if ids[idx] == end_id:
            return -1
    return -1


class ThinkingBudgetLogitProcessor(CustomLogitProcessor):
    """控制 thinking 长度的 logit processor 基类。"""

    THINKING_START_TOKEN_ID: int
    THINKING_END_TOKEN_ID: int
    NEW_LINE_TOKEN_ID: int

    def __call__(self, logits, custom_param_list: list[dict[str, Any]]):
        if custom_param_list is None or not custom_param_list:
            return logits
        for i, param_dict in enumerate(custom_param_list):
            if param_dict is None:
                continue

            thinking_budget: int | None = param_dict.get("thinking_budget")
            # 未设置、非整数或负数时直接跳过，不干预采样
            if (
                thinking_budget is None
                or not isinstance(thinking_budget, int)
                or thinking_budget < 0
            ):
                continue
            req: Req = param_dict.get("__req__")
            cur_ids: list[int] = [*req.origin_input_ids, *req.output_ids]

            # 关键修复：原来用 `THINKING_START not in cur_ids or THINKING_END in cur_ids`
            # 判定块状态，对 end token 出现在 prompt 中的模型（如 Inkling）永远视为
            # " 已关闭 "，导致 budget 被静默忽略。现在只认 " 最近一个未闭合的 start"。
            start_index = _open_thinking_start(
                cur_ids, self.THINKING_START_TOKEN_ID, self.THINKING_END_TOKEN_ID
            )
            if start_index < 0:
                continue

            # 从最近一个打开的 thinking start 开始计数
            num_tokens_after_start = len(cur_ids) - start_index - 1
            if num_tokens_after_start < thinking_budget:
                continue

            # 预算耗尽：先强制换行（对齐推理文本格式），下一步再强制 end token
            if not req.output_ids or req.output_ids[-1] != self.NEW_LINE_TOKEN_ID:
                logits[i, :] = -float("inf")
                logits[i, self.NEW_LINE_TOKEN_ID] = 0.0
                continue
```

```
            logits[i, :] = -float("inf")
            logits[i, self.THINKING_END_TOKEN_ID] = 0.0

        return logits


class InklingThinkingBudgetLogitProcessor(ThinkingBudgetLogitProcessor):
    """Inkling 模型的 thinking budget 控制。

    Inkling 的 END_MESSAGE token（200010）同时充当消息终止符，因此 prompt
    中必然反复出现 end token，这正是基类旧逻辑失效的根源；该子类仅需声明
    三个 token ID，不做额外逻辑。
    """

    THINKING_START_TOKEN_ID: int = 200008
    THINKING_END_TOKEN_ID: int = 200010
    NEW_LINE_TOKEN_ID: int = 198
```

评论区精华

JustinTong0323 的 review 认可 delimiter 修复方向（"The delimiter fix looks correct"），但要求补充 Inkling 子类的直接测试覆盖："Exercise InklingThinkingBudgetLogitProcessor directly here, assert its framing IDs against inkling_tokenizer.py, and cover the forced newline/end path; the current Qwen3-based test leaves the new Inkling constants untested." 作者随后确认已在 563c6672bb 提交中按建议补充回归测试，最终 APPROVED / LGTM，无未解决的疑虑。

- 为 Inkling 子类补充直接测试覆盖 (testing): 作者在 563c6672bb 提交中按建议补充了 test_inkling_end_token_in_prompt_does_not_disable_budget，直接实例化 Inkling 子类、断言 token ID 与 tokenizer 常量一致，并覆盖强制 newline 与强制 end 两条路径。
- 基类 delimiter 判定修复的正确性 (correctness): 评审通过，无未解决疑虑。

风险与影响

- 风险：基类语义变化影响所有 ThinkingBudgetLogitProcessor 子类（GLM-4.5 / Qwen3 / DeepSeek-R1 / Inkling）：计数语义从 " 第一个 start" 变为 " 最近一个未闭合 start"。对单 delimiter 序列行为不变（现有测试全过），但含多个 thinking 块的序列行为会改变，属修复而非回归，仍需留意线上多块输出的场景。热路径开销方面，__call__ 在每步 decode 都会执行，反向全扫最坏 O(n)，原实现两次 in 加一次 index 同样是 O(n)，无实质恶化，但长序列下单步扫描成本仍存在。token ID 硬编码风险：Inkling 的 200008 / 200010 通过测试断言与 inkling_tokenizer.py 的 INKLING_SPECIAL_TOKEN_IDS 绑定，tokenizer 演进时需同步修改两处。
- 影响：对用户：Inkling 模型服务的用户现在可以设置 thinking_budget 控制思考长度，实测超预算率从 8/8 降至 0/8，且不影响回答正确性；GLM / Qwen / DeepSeek 用户不受影响（单 delimiter 行为不变），但含多 thinking 块场景的语义更准确。对系统：采样热路径新增一个反向扫描函数，性能开销与原有逻辑同级，无需额外资源。对团队：为 custom logit

processor 家族新增一个模型变体样例，并修复了基类设计缺陷，后续接入 "end token 出现在 prompt 中 " 的模型（消息级终止符）不会再踩这个坑。

- 风险标记：核心采样热路径变更，影响多模型基类语义，token ID 硬编码依赖

关联脉络

- PR #34250 Update dspark draft path in Inkling small cookbook: 同为 Inkling-Small 模型支持线：该 PR 更新 cookbook 中的 DSPARK draft 路径，本 PR 为 Inkling 增加 thinking budget 能力，二者共同完善 Inkling 模型在 SGLang 中的推理与部署体验。
- PR #11416 thinking budget 原始实现（PR body 引用，标题未在材料中提供）：PR body 中作者请曾评审 #11416 的 JustinTong0323 复审，说明本 PR 是对 #11416 引入的 thinking budget 机制的扩展与基类修复。