

PR #38693 完整报告

sgl-project/sglang

Add granite_thinking_parser reasoning parser for Granite 4.2

合并时间: 2026-09-11 22:18

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38693>

执行摘要

- 一句话: 新增 Granite 4.2 思考解析器与模板自动检测规则
- 推荐动作: 值得精读, 尤其是想新增模型 reasoning parser 的工程师:
GraniteThinkingDetector 是「薄覆写基类检测器」的典型范例, 展示如何在不动核心解析逻辑的前提下实现模型特化行为。建议重点看三处设计决策——(1) 用 content_absent 显式区分「正文缺失」与「正文仅含换行」的 swap 语义; (2) 流式剥离为何必须同时采样调用前后状态才能做到分块无关; (3) 模板判别为何选 defer_loading 而非更显眼的 <parameter= 工具块。另外建议沿提交 47dd55e 阅读被修复的三个 bug, 理解自动检测规则的误判成本; 测试裁剪的取舍 (删除伪 parity 测试) 也值得作为测试设计案例讨论。

功能与动机

Granite 4.2 的 chat template 在推理与正文边界处写的是 `\n</think>\n`, 普通 DeepSeek-R1 风格的解析器会把前导换行带进 content; 同时 `enable_thinking=False` 或输出被 max_tokens 截断时会出现正文为空的情况。此前用户必须挂载 HF 插件才能正确解析, PR body 明确目标是「enabling `--reasoning-parser granite_thinking_parser` without the HF plugin」, 并让 parser 名称与 vLLM PR#55957 保持一致以实现跨引擎对齐, 行为上对齐 HF 插件 `granite_thinking_parser.py`。

实现拆解

1. 新增检测器: python/sglang/srt/parser/reasoning_parser.py 中新增 GraniteThinkingDetector, 继承 BaseReasoningFormatDetector, 沿用 `<think>/</think>` 标记、`<tool_call>` 工具起始标记、`reasoning_default="enable_thinking"`。非流式 detect_and_parse 在基类解析结果上做两步后处理: 先判断 `</think>` 是否存在与正文是否为空 (content_absent), 对存在的正文 `rstrip("\n")`; 再在 `force_nonempty_content` 且正文为空且 (正文确实缺失或根本没有 `</think>`) 时把 reasoning_text 与 normal_text 对调。
2. 注册与参数映射: DetectorMap 新增 "granite_thinking_parser": GraniteThinkingDetector; ReasoningParser.__init__ 新增分支, 当 `chat_template_kwargs["enable_thinking"] is False` 时自动注入 `force_nonempty_content=True`, 把「不思考」语义直接映射为「内容不得为空」的解析行为, 避免上层调用方各自处理。

3. 流式状态机：新增 `parse_streaming_increment` 覆写，用 `_reasoning_seen`（跨 chunk 累积「见过推理段」的证据）和 `_content_started`（正文是否已经开始）控制只在正文起点剥离一次换行，正文开始后换行原样保留。`_reasoning_seen` 的证据来源同时包含调用前的 `_in_reasoning`、调用后的 `_in_reasoning`、`stripped_think_start` 和本次返回的 `reasoning_text`，保证整个 `think` 块落在单个 chunk 时也不会漏判。
4. 模板自动检测：`python/sglang/srt/parser/template_detection.py` 新增谓词 `_is_granite_thinking_parser`，判据为 `truncate_history_thinking + defer_loading + reasoning_config` 的 `enable_thinking/` 默认开启，并在 `REASONING_PARSER_RULES` 中注册在 `nemotron_3` 之前。关键点是放弃用 `<parameter=` 工具调用块做判别——Granite 4.2 与 Nemotron-3 模板逐字共享该块。
5. 测试与裁剪：`test/registered/unit/parser/test_reasoning_parser.py` 新增 `TestGraniteThinkingDetector`（含非流式、流式、`ReasoningParser` 集成、`enable_thinking=False` 映射用例）；`test/registered/unit/parser/test_template_manager.py` 新增两条模板判别测试（Nemotron-3 不误判为 Granite、Granite 经 `defer_loading` 命中）。首版提交中的 `test_granite_thinking_parity.py` 在后续提交中被删除，理由是固定用例与检测器测试重复且硬编码期望无法证明与 HF 插件真实一致。

关键文件：

- `python/sglang/srt/parser/reasoning_parser.py`（模块 推理解析器；类别 `source`；类型 `core-logic`；符号 `GraniteThinkingDetector`, `init`, `detect_and_parse`, `parse_streaming_increment`）：核心变更文件：新增 `GraniteThinkingDetector` 类（含非流式 `swap/lstrip` 与流式状态机），注册 `DetectorMap`，并在 `ReasoningParser.__init__` 中把 `enable_thinking=False` 映射为 `force_nonempty_content`。
- `python/sglang/srt/parser/template_detection.py`（模块 模板检测；类别 `source`；类型 `core-logic`；符号 `_is_granite_thinking_parser`, `REASONING_PARSER_RULES`）：新增模板自动检测谓词 `_is_granite_thinking_parser` 并在 `REASONING_PARSER_RULES` 中注册，用 `defer_loading` 与 Nemotron-3 区分，是 `auto` 模式正确路由的关键。
- `test/registered/unit/parser/test_reasoning_parser.py`（模块 解析器测试；类别 `test`；类型 `test-coverage`；符号 `TestGraniteThinkingDetector`, `setUp`, `test_leading_newline_stripped`, `test_force_nonempty_no_swap_when_think_end_present`）：新增 `TestGraniteThinkingDetector`，覆盖非流式 `swap` 语义、流式换行剥离、分块无关性与 `ReasoningParser` 集成，是本次行为契约的主要护栏。
- `test/registered/unit/parser/test_template_manager.py`（模块 模板测试；类别 `test`；类型 `test-coverage`；符号 `test_nemotron_with_shared_parameter_block_not_misclassified_as_granite`, `test_granite_detected_via_defer_loading_signature`）：新增两条模板判别回归测试：Nemotron-3 共享 `<parameter=` 块时不被误判为 Granite，Granite 通过 `defer_loading` 命中新规则。

关键符号：`GraniteThinkingDetector.init`, `GraniteThinkingDetector.detect_and_parse`, `GraniteThinkingDetector.parse_streaming_increment`, `ReasoningParser.init`, `_is_granite_thinking_parser`, `test_streaming_result_is_chunking_independent`, `test_granite_detected_via_defer_loading_signature`

关键源码片段

python/sglang/srt/parser/reasoning_parser.py

核心变更文件：新增 GraniteThinkingDetector 类（含非流式 swap/lstrip 与流式状态机），注册 DetectorMap，并在 ReasoningParser.__init__ 中把 enable_thinking=False 映射为 force_nonempty_content。

```
class GraniteThinkingDetector(BaseReasoningFormatDetector):
    """Granite 4.2 thinking 模型（ibm-granite/granite-4.2-*）专用检测器。

    Granite chat template 在边界处写入 ``\n</think>\n``，因此正文以换行开头，
    需要剥离；行为对齐 HF 插件 granite_thinking_parser.py。
    """

    def __init__(
        self,
        stream_reasoning: bool = True,
        force_reasoning: bool = False,
        continue_final_message: bool = False,
        previous_content: str = "",
        force_nonempty_content: bool = False,
    ):
        super().__init__(
            "<think>",
            "</think>",
            tool_start_token="<tool_call>", # 与 Nemotron-3 共用，因此不能作为模板判据
            reasoning_default="enable_thinking", # 与 Granite 模板的 toggle 参数一致
            force_nonempty_content=force_nonempty_content,
            # ... 其余参数（stream/force/continue_final_message）同基类直传
        )
        self._content_started = False # 正文是否已经开始：只在起点剥离一次换行
        self._reasoning_seen = False # 是否见过推理段：跨 chunk 累积的判据

    def detect_and_parse(self, text: str) -> StreamingParseResult:
        """非流式解析：先继承基类切分，再施加两条 Granite 特化行为。"""
        ret = self._detect_and_parse_impl(text)
        think_end_present = self.think_end_token in text
        # 注意同时要求「有 </think>」且「正文为空」，才认定为正文确实缺失；
        # 仅含换行的正文不属于此列，交由下面的 lstrip 处理而不参与 swap。
        content_absent = think_end_present and not ret.normal_text
        if think_end_present and ret.normal_text:
            ret.normal_text = ret.normal_text.lstrip("\n")
        if (
            self._force_nonempty_content
            and not ret.normal_text
            and (content_absent or not think_end_present)
        ):
            # enable_thinking=False 或输出被截断时，把推理内容反转为正文，
```

```

        # 避免向用户返回空 content
        ret.normal_text, ret.reasoning_text = ret.reasoning_text, ret.normal_text
    return ret

def parse_streaming_increment(self, new_text: str) -> StreamingParseResult:
    """流式解析：保证剥离结果与 chunk 切分粒度无关。"""
    was_in_reasoning = self._in_reasoning
    ret = super().parse_streaming_increment(new_text)
    # 同时采样调用前状态（was_in_reasoning）、调用后状态（_in_reasoning）、
    # stripped_think_start 与本次生成的 reasoning_text——只采样调用前状态
    # 会让「整个 think 块落在同一个 chunk」时漏判，导致前导换行泄漏。
    if (
        was_in_reasoning
        or self._in_reasoning
        or self.stripped_think_start
        or ret.reasoning_text
    ):
        self._reasoning_seen = True
    if self._reasoning_seen and not self._content_started and ret.normal_text:
        ret.normal_text = ret.normal_text.lstrip("\n")
    if ret.normal_text:
        self._content_started = True # 正文一旦开始，后续换行原样保留
    return ret

```

python/sglang/srt/parser/template_detection.py

新增模板自动检测谓词 `_is_granite_thinking_parser` 并在 `REASONING_PARSER_RULES` 中注册，用 `defer_loading` 与 `Nemotron-3` 区分，是 `auto` 模式正确路由的关键。

```

def _is_granite_thinking_parser(ctx):
    # Granite 4.2 与 Nemotron-3 的模板逐字共享 <function=name>/<parameter=key>
    # 工具调用说明块，因此它无法用来区分；defer_loading 是 Granite 独有的
    # 延迟工具加载特征，作为唯一判别信号。
    return (
        ctx.has_text("truncate_history_thinking") # 与 Nemotron-3 共有的推理历史裁剪
        and ctx.has_text("defer_loading") # Granite 独有：延迟工具加载
        and ctx.reasoning_config is not None
        and ctx.reasoning_config.toggle_param == "enable_thinking"
        and ctx.reasoning_config.default_enabled is True
    )

```

规则按顺序匹配，Granite 规则必须排在 `nemotron_3` 之前：

两者共享 `truncate_history_thinking` 特征，靠更严格的 `defer_loading` 抢先命中。

```

REASONING_PARSER_RULES = (
    DetectionRule(name="k2_horizon", value="k2_horizon", predicate=_is_k2_v3),
    # ... 前置规则省略
    DetectionRule(name="kimi_k2", value="kimi_k2", predicate=_is_kimi_k2),
    DetectionRule(
        name="granite_thinking_parser",

```

```

        value="granite_thinking_parser",
        predicate=_is_granite_thinking_parser,
    ),
    DetectionRule(name="nemotron_3", value="nemotron_3", predicate=_is_nemotron_3),
    # ... 后续规则省略
)

```

test/registered/unit/parser/test_reasoning_parser.py

新增 TestGraniteThinkingDetector，覆盖非流式 swap 语义、流式换行剥离、分块无关性与 ReasoningParser 集成，是本次行为契约的主要护栏。

```

class TestGraniteThinkingDetector(CustomTestCase):
    def setUp(self):
        self.detector = GraniteThinkingDetector()

    def test_force_nonempty_no_swap_when_think_end_present(self):
        """</think> 存在时即使剥离后正文为空也不 swap（仅换行不算正文缺失），
        与 HF 插件行为一致。"""
        detector = GraniteThinkingDetector(force_nonempty_content=True)
        result = detector.detect_and_parse("<think>reasoning</think>")

    def test_force_nonempty_swaps_when_text_ends_at_think_end(self):
        """正文确实缺失（例如 max_tokens 恰好截在 </think>）时执行 swap。"""
        detector = GraniteThinkingDetector(force_nonempty_content=True)
        result = detector.detect_and_parse("<think>reasoning</think>")
        self.assertEqual(result.reasoning_text, "")
        self.assertEqual(result.normal_text, "reasoning")

    def test_streaming_result_is_chunking_independent(self):
        # 同一段文本在不同切分粒度（整段 / 两段 / 三段 / 逐字符）下，
        # 流式累积输出必须完全一致，这是流式剥离逻辑的核心回归护栏。
        # 空 think 块这一组只触发 stripped_think_start 证据：
        # 此时 reasoning_text 为空、调用前后 _in_reasoning 均为 False。
        for text, exp_r, exp_c in (
            ("<think>r</think>\nHello", "r", "Hello"),
            ("<think></think>\nHello", "", "Hello"),
        ):
            for stream_reasoning in (True, False):
                for chunks in (
                    [text],
                    [text[: text.index("</think>") + len("</think>")], "\nHello"],
                    ["<think>", text[len("<think>") : text.index("</think>")], "</think>", "\nHello"],
                    list(text),
                ):
                    with self.subTest(text=text, stream_reasoning=stream_reasoning, chunks=chunks)

```

```

:
    detector = GraniteThinkingDetector(stream_reasoning=stream_reasoning)
    all_r = all_c = ""
    for chunk in chunks:
        ret = detector.parse_streaming_increment(chunk)
        all_r += ret.reasoning_text
        all_c += ret.normal_text
    end = detector.finish()
    all_r += end.reasoning_text
    all_c += end.normal_text
    self.assertEqual(all_r, exp_r)
    self.assertEqual(all_c, exp_c)

def test_enable_thinking_false_swaps_truncated_reasoning(self):
    """经 ChatCompletionRequest 的 chat_template_kwargs 注入
    enable_thinking=False 后，ReasoningParser 应自动启用内容非空回退。"""
    from sglang.srt.entrypoints.openai.protocol import ChatCompletionRequest

    request = ChatCompletionRequest(
        model="granite-4.2-30b",
        messages=[{"role": "user", "content": "hi"}],
        chat_template_kwargs={"enable_thinking": False},
    )
    parser = ReasoningParser("granite_thinking_parser", request=request)
    reasoning, normal = parser.parse_non_stream("<think>truncated")
    self.assertEqual(reasoning, "")
    self.assertEqual(normal, "truncated")

```

评论区精华

讨论集中在维护者 JustinTong0323 推入的修复提交 47dd55e (review 中评论: 「Pushed a fix commit (47dd55e) addressing three issues found in review」) :

「Auto-detection misrouted Nemotron-3 — `<parameter=` can't discriminate: the Granite 4.2 and Nemotron-3 templates share that tool-call block verbatim, so `--reasoning-parser auto` switched Nemotron deployments to the Granite detector. Now uses `defer_loading`」

「Empty-content fallback missed the end-tag boundary — with `force_nonempty_content`, text ending exactly at `</think>` returned empty content; the HF plugin swaps reasoning→content whenever parsed content is absent.」

「Streaming newline strip was chunking-dependent — `_reasoning_seen` sampled only pre-call state, so a whole think block in one chunk leaked the leading newline.」

他声称用官方模板逐一验证: Granite 4.2 3B/8B/30B/MLX 命中 `granite_thinking_parser`, Nemotron-3 Nano/Super/Ultra 命中 `nemotron_3`, 并提出 79/79 独立验证检查。后续两个提交 (4ddd9da、44005a6) 系统性削减测试, 理由是 「its fixed cases duplicate TestGraniteThinkingDetector… hardcoded expectations cannot prove live HF plugin

parity」，即不保留无法证明真实对齐的伪 parity 测试。最终 review 结论为 APPROVED「LGTM」，并以 `/tag-and-rerun-ci` 触发重跑。

- 模板自动检测误把 Nemotron-3 路由到 Granite (correctness): 改用 `defer_loading` 作为唯一判别信号，规则顺序排在 `nemotron_3` 之前；补充两条互斥性回归测试。
- `force_nonempty_content` 在边界处漏掉 `swap (correctness)`: 引入 `content_absent = think_end_present and not ret.normal_text` 作为 `swap` 的额外条件，只换行内容仍保持空正文；新增 `test_force_nonempty_swaps_when_text_ends_at_think_end` 与 `test_force_nonempty_no_swap_when_think_end_present` 锁定两种语义。
- 流式换行剥离依赖 chunk 切分粒度 (correctness): 证据来源扩展到调用后 `_in_reasoning`、`stripped_think_start` 和本次 `reasoning_text`，并新增多组切分粒度的 `chunking-independence` 测试（覆盖 `stream_reasoning True/False` 与逐字符切分）。
- 是否保留 granite parity 测试与冗余单测 (testing): 删除 parity 测试文件，保留唯一的 `enable_thinking=False` 映射检查并改写成通过 `ChatCompletionRequest` 断言最终输出；其余按「一个失败模式一个用例」原则精简。结果是本 PR 净增 -0 删除但测试从更宽收敛到更精准。
- `enable_thinking=False` 的参数映射位置 (design): 在 `ReasoningParser.__init__` 中以 `model_type.lower() == "granite_thinking_parser"` 分支注入 `force_nonempty_content=True`；该分支目前没有配独立的 `--reasoning-parser auto` 端测试，仅由单测通过 `ChatCompletionRequest` 覆盖。

风险与影响

- 风险：

1. 自动检测规则顺序敏感：`_is_granite_thinking_parser` 与 `_is_nemotron_3` 共享 `truncate_history_thinking` 核心特征，仅靠 `defer_loading` 区分。若后续模型模板引入 `defer_loading` 字段，或 Nemotron 系模板加入该关键字，`--reasoning-parser auto` 可能再次误路由，属于隐性耦合。
2. 流式状态机新增可变状态：`_content_started/_reasoning_seen` 在检测器实例生命周期内只前进不回退，`continue_final_message` 多轮续写场景复用同一实例时行为需额外验证；现有测试仅覆盖单次请求。
3. 剥离范围较窄：只 `lstrip("\n")`，不处理空格或 `

；若 Granite 后续模板或不同 tokenizer 产出

` 会遗漏，属于与 HF 插件语义绑定的取舍。

1. 对 HF 插件的对齐缺少自动化护栏：原 `test_granite_thinking_parity.py` 被删除后，对齐性只靠人工比对与注释声明，HF 插件侧行为变更不会触发本仓库测试失败。
2. CI 状态不完整：PR body 中 Extra 与 AMD ROCm 两个测试 job 显示为失败 (x:)，最终在 Base job 进行中即被合并，未留下全绿记录。- 影响：对用户：Granite 4.2 thinking 用户可直接使用 `--reasoning-parser granite_thinking_parser`，无需 HF 插件，且与 vLLM 端命名一致，迁移成本降低；auto 模式下的 Granite 部署能自动命中正确 parser。对系统：变更集中在 parser 层，不触及调度、KV cache、注意力等核心路径，回归面可控；但 `REASONING_PARSER_RULES` 的判别顺序属于全局共享逻辑，所有使用

auto 的模型部署都受其影响，Nemotron-3 用户是本次修复的直接受益方。对 团队：本次 PR 展示了「新增检测器 + 自动检测谓词 + 单测」的标准扩展范式，可作为后续新模型 reasoning parser 的模板；同时维护者直接在贡献者分支上推修复提交，说明外部贡献与核心团队的协作路径高效但要求 reviewer 具备交叉模板的领域知识。- 风险标记：自动检测规则顺序敏感，流式状态机新增可变状态，HF 插件对齐无自动化验证，CI 未全绿即合入

关联脉络

- PR #38297 Auto-detect GLM-5.3 chat templates as glm45/glm47 parsers: 同样在 `python/sglang/srt/parser/template_detection.py` 中新增模板自动检测规则并补充 `test_template_manager.py` 用例，属于同一条「按模板特征自动选 parser」的功能线，本次的 `_is_granite_thinking_parser` 与其在判别策略（结构特征互斥）上高度同源。
- PR #38993 [Refactor] Generalize attention graph variants in the decode runner: 同为仓库内 `parser/` 契约抽取类的结构性整理，虽模块不同，但都体现「把模型特化差异收敛为可注册契约」的演进思路，可作为对照阅读。