

PR #56635 完整报告

vllm-project/vllm

[Parser] Fix: correct parser frontend handling of reasoning end and boundary tokens

合并时间: 2026-09-16 16:17

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56635>

执行摘要

- 一句话: 修复 parser 基类 reasoning 结束与回合边界的 token 扫描, GLM 声明边界 token
- 推荐动作: 值得精读, 尤其是 `ParserEngine.extract_content_ids` 从「反向取最后一个 marker」改为「先定位回合边界、再正向取首个结束符」这一段——它同时修正了内容丢弃与多轮误判两类 bug, 是理解 vLLM declarative parser 状态机的关键切入点。建议重点关注三处设计决策: 一是把多结束符语义集中到基类、模型侧仅声明 `turn_boundary_tokens`; 二是用 `wait_for_reasoning` (并回退到 `initial_state`) 作为「是否需要裁剪」的唯一判据, 取代子类 `override`; 三是 gaby 给出的「reasoning-end 优先、tool marker 兜底」优先级排序。落地后建议为 ling3 等 GLM 派生 parser 补充同类边界测试, 并考虑增加一条真实 chat template 的端到端用例, 降低基类改动带来的隐性回归风险。

功能与动机

GLM-4.7 在 thinking 模式下可以不经 `</think>` 直接输出 `<tool_call>`, `<tool_call>` 实质上是隐式 reasoning 终止符, 而基类 `ParserEngine.extract_content_ids` 只认 `</think>`。PR body 明确列出两个 bug: 「`extract_content_ids` drops content on mixed deltas containing both tool calls and think ends」与「`is_reasoning_end` can incorrectly report True for a fresh turn without boundary guards」。作者在评论中补充动机是修复 #56605 暴露的 token bleed; review 中 gaby 指出现有实现无法修复该问题并给出反例后, 作者撤回该声明, 把 PR 收敛为 parser frontend 自身的边界处理与基类集中化。

实现拆解

1. 基类补齐多结束符 + 回合边界扫描 (`vllm/parser/engine/parser_engine.py`):
`ParserEngine.extract_content_ids` 新增三段逻辑——先解析 `config.wait_for_reasoning` (为 `None` 时回退到 `config.initial_state is ParserState.REASONING`);
`wait_for_reasoning` 为假时直接返回 `input_ids`, 使 thinking 关闭的 parser 不再被裁剪; 单结束符场景保留原「最后一个 `</think>` 之后」语义, 多结束符场景先用 `_turn_boundary_token_ids` 反向定位当前轮起点 `turn_start`, 再正向扫描首个 `_reasoning_end_token_ids` 成员并返回 `input_ids[i:]`。原因: 反向扫描「谁在最后就取谁」会把 `</think>` 与后续 tool call 之间的正文静默丢掉 (gaby 给出的反例 [`<think>`, 10, 11, `</think>`, 20, 21, `<tool_call>`, 30, `</tool_call>`])。影响: `extract_content_ids` 是 `abstract_parser.py` 调用的公共入口, 所有继承 `ParserEngine` 的 parser 都受此行为变化

影响。

2. GLM-4.7 改为声明式回合边界 (vllm/parser/glm47_moe.py) : 新增
GLM_TURN_BOUNDARIES = frozenset(("<|system|>", "<|user|>", "<|assistant|>", "<|observation|>")) 并传入 ParserEngineConfig.turn_boundary_tokens, 让基类在反向扫描时能识别「对话轮」边界, 避免把历史轮的 </think> / <tool_call> 当作当前轮状态; 同时删除 config 中手写的 wait_for_reasoning=thinking, 因为
initial_state=ParserState.REASONING if thinking else ParserState.CONTENT 已能表达同一语义, 基类会自行推导。
3. 删除模型侧 override, 语义上移 (vllm/parser/glm47_moe.py) : 移除
Glm47MoeParser.extract_content_ids (原实现是 if not self.thinking_enabled: return input_ids), 该分支现由基类的 wait_for_reasoning 判断统一承担; extract_reasoning 仍保留在子类, 因为它表达的是模型特有的「thinking 关闭时不拆分 reasoning」语义。收益是 ling3 等继承 GLM parser 的模型自动获得同样的边界修复。
4. 测试配套 (tests/parser/engine/test_glm47_moe.py, 新增 +108) : 用
tests.parser.engine.conf.test.make_mock_tokenizer 构造 7 项词表 (think / tool call / 三种回合边界 token / 普通文本 token), 分两个测试类固化契约。TestIsReasoningEnd 覆盖 open reasoning、显式 </think>、无 </think> 直接工具调用、上一轮 tool call 与 think end 被忽略、空输入、thinking 关闭; TestExtractContentIds 覆盖 </think> 优先于后续 tool call、</think> 后多个 tool call 全保留、无 </think> 时回退到第一个 tool call、上一轮 tool call 被忽略、无 markers 原样返回、thinking 关闭原样返回。
5. 配置与部署配套: 无 schema、API、部署脚本改动; 唯一新增的配置项是
ParserEngineConfig.turn_boundary_tokens 的使用方 (GLM), 因此其它多结束符 parser 若未声明边界 token, turn_start 会退化为 0, 需要各自评估。

关键文件:

- vllm/parser/engine/parser_engine.py (模块 解析引擎; 类别 source; 类型 core-logic; 符号 ParserEngine.extract_content_ids, ParserEngine.is_reasoning_end) : PR 的核心修复点: 基类 extract_content_ids 增加 wait_for_reasoning 早返回、多结束符扫描与回合边界定位, 直接决定所有继承 ParserEngine 的模型如何剥离 reasoning 前缀。
- vllm/parser/glm47_moe.py (模块 GLM 解析器; 类别 source; 类型 configuration; 符号 GLM_TURN_BOUNDARIES, glm47_moe_config, Glm47MoeParser.extract_content_ids, Glm47MoeParser.extract_reasoning) :
GLM-4.7 侧的声明式改造: 新增 GLM_TURN_BOUNDARIES 并接入 turn_boundary_tokens, 删除 wait_for_reasoning=thinking 与 Glm47MoeParser.extract_content_ids override, 是「模型侧只声明、基类做逻辑」范式的落点。
- tests/parser/engine/test_glm47_moe.py (模块 解析测试; 类别 test; 类型 test-coverage; 符号 TestIsReasoningEnd, TestExtractContentIds, test_tool_call_without_think_end, test_previous_turn_tool_call_ignored) : 新增的 token 级单测, 把「无 </think> 时工具调用即隐式 reasoning 结束」「上一轮标记必须被忽略」「thinking 关闭不裁剪」等契约固化为可回归断言, 是本次行为变更的唯一自动化防线。

关键符号：ParserEngine.extract_content_ids, ParserEngine.is_reasoning_end, glm47_moe_config, Glm47MoeParser.extract_content_ids, Glm47MoeParser.extract_reasoning, TestIsReasoningEnd.test_tool_call_without_think_end, TestExtractContentIds.test_think_end_wins_over_later_tool_call

关键源码片段

vllm/parser/engine/parser_engine.py

PR 的核心修复点：基类 `extract_content_ids` 增加 `wait_for_reasoning` 早返回、多结束符扫描与回合边界定位，直接决定所有继承 ParserEngine 的模型如何剥离 reasoning 前缀。

```
# vllm/parser/engine/parser_engine.py —— ParserEngine.extract_content_ids
# 作用：从 token 序列里剥掉 reasoning 前缀，返回应交给 tool parser / 前端的内容。
def extract_content_ids(self, input_ids: list[int]) -> list[int]:
    config = self.parser_engine_config
    wait_for_reasoning = config.wait_for_reasoning
    # 未显式声明时按 initial_state 推导：从 REASONING 起步就意味着要等 reasoning 结束。
    # 这一回退让 GLM 可以删掉 config 里手写的 wait_for_reasoning=thinking。
    if wait_for_reasoning is None:
        wait_for_reasoning = config.initial_state is ParserState.REASONING
    # thinking=False 的 parser 不应裁剪任何 token，原样返回（原先由 Glm47MoeParser
    # 的 override 承担，现已上移到基类）。
    if not wait_for_reasoning:
        return input_ids

    end_id = self._reasoning_end_token_id
    if end_id is not None:
        # 单结束符路径：取「最后一个」 </think> 之后的所有 token。
        # 这样做才能保住 </think> 与后续 tool call 之间的正文，
        # 而不是像旧实现那样只保留最后一个 marker 之后的内容。
        for i in range(len(input_ids) - 1, -1, -1):
            if input_ids[i] == end_id:
                return input_ids[i + 1 :]

    end_ids = self._reasoning_end_token_ids
    if end_ids:
        turn_start = 0
        boundary_ids = self._turn_boundary_token_ids
        # 多结束符场景（GLM 的 </think> 与 <tool_call> 都是隐式 reasoning 结束符）：
        # 先反向定位「当前轮」起点，上一轮的 </think> / <tool_call> 不能算当前轮状态。
        # 边界由模型侧通过 turn_boundary_tokens 声明，未声明时 turn_start 退化为 0。
        for i in range(len(input_ids) - 1, -1, -1):
            if input_ids[i] in boundary_ids:
                turn_start = i + 1
                break
        # 再正向扫描首个结束符，并保留其后的全部内容：
        # 既覆盖「无 </think> 直接进 tool call」，也保留同轮出现的多个 tool call。
        for i in range(turn_start, len(input_ids)):
```

```

        if input_ids[i] in end_ids:
            return input_ids[i:]
# 完全没有命中结束符：不做裁剪，交给上层逻辑处理。
return input_ids

```

vllm/parser/glm47_moe.py

GLM-4.7 侧的声明式改造：新增 `GLM_TURN_BOUNDARIES` 并接入 `turn_boundary_tokens`，删除 `wait_for_reasoning=thinking` 与 `Glm47MoeParser.extract_content_ids override`，是「模型侧只声明、基类做逻辑」范式的落点。

```

# vllm/parser/glm47_moe.py —— GLM-4.7 的回合边界声明与 parser 配置
# 渲染后的 GLM prompt 用这些特殊 token 切分对话回合。上一轮的 reasoning 标记
# 不能当成「当前正在生成的那一轮」的状态，否则 fresh turn 会被误判。
GLM_TURN_BOUNDARIES = frozenset(
    ("<|system|>", "<|user|>", "<|assistant|>", "<|observation|>")
)

```

```

@functools.cache
def glm47_moe_config(thinking: bool = True) -> ParserEngineConfig:
    # 此处省略 reasoning_terminals / arg_tag_transitions 等定义
    return ParserEngineConfig(
        name="glm47_moe",
        # thinking 决定初始状态；基类在 wait_for_reasoning 为 None 时会从这个字段推导，
        # 因此不再需要手写 wait_for_reasoning=thinking（本次删除）。
        initial_state=ParserState.REASONING if thinking else ParserState.CONTENT,
        terminals={
            **reasoning_terminals,
            "TOOL_START": TOOL_CALL_START,
            "TOOL_END": TOOL_CALL_END,
            # ARG_KEY_* / ARG_VALUE_* 等省略
        },
        token_id_terminals={
            **reasoning_token_id_terminals, # THINK_START / THINK_END
            # <tool_call> 同时注册为 token 级终结符，也就是隐式 reasoning 结束符
            "TOOL_START": TOOL_CALL_START,
            "TOOL_END": TOOL_CALL_END,
        },
        transitions={
            **reasoning_transitions,
            (ParserState.REASONING, "TOOL_START"): Transition(
                ParserState.TOOL_NAME,
                # 从 reasoning 直接跳到工具调用：一次事件同时结束 reasoning 并开工具
                (EventType.REASONING_END, EventType.TOOL_CALL_START),
            ),
            (ParserState.CONTENT, "TOOL_START"): Transition(
                ParserState.TOOL_NAME,
                (EventType.TOOL_CALL_START,),
            ),
        }
    )

```

```

        # TOOL_NAME / TOOL_ARGS 相关转移及 arg_tag_transitions 省略
    },
    # 本次关键新增：把回合边界交给基类，供 extract_content_ids / is_reasoning_end
    # 在反向扫描时重新定位「当前轮」的起点。
    turn_boundary_tokens=GLM_TURN_BOUNDARIES,
    arg_converter=_glm47_arg_converter,
    stream_arg_deltas=True,
    tool_args_json=False,
    validate_tool_names=True,
)

```

```

class Glm47MoeParser(ParserEngine):
    # 变更说明：原先这里的 extract_content_ids override
    # (thinking 关闭时直接 return input_ids) 已被删除，同一语义改由基类的
    # wait_for_reasoning 分支统一承担；extract_reasoning 仍保留在子类，
    # 因为它表达的是模型特有的「不拆分 reasoning、整体作为 content」行为。
    def extract_reasoning(self, model_output, request):
        if not self.thinking_enabled:
            return None, model_output
        return super().extract_reasoning(model_output, request)

```

tests/parser/engine/test_glm47_moe.py

新增的 token 级单测，把「无 `</think>` 时工具调用即隐式 reasoning 结束」「上一轮标记必须被忽略」「thinking 关闭不裁剪」等契约固化为可回归断言，是本次行为变更的唯一自动化防线。

tests/parser/engine/test_glm47_moe.py —— 关键测试契约（节选）

```

class TestExtractContentIds:
    def test_think_end_wins_over_later_tool_call(self, parser):
        """``</think>`` 之后的正文不能因为后面还有 tool call 就被丢弃。

        这是 gaby 在 review 中给出的反例：旧的反向扫描只返回最后一个 marker
        之后的内容，会把 20、21 静默吃掉。
        """
        ids = [THINK_S, TEXT, THINK_E, 20, 21, TOOL_S, 30, TOOL_E]
        assert parser.extract_content_ids(ids) == [20, 21, TOOL_S, 30, TOOL_E]

    def test_every_tool_call_after_think_end_kept(self, parser):
        # </think> 之后出现的多个 tool call 必须全部保留（正向扫描的结果）。
        ids = [THINK_E, 20, TOOL_S, 30, TOOL_E, 21, TOOL_S, 31, TOOL_E]
        assert parser.extract_content_ids(ids) == ids[1:]

    def test_falls_back_to_tool_call(self, parser):
        """没有 ``</think>`` 时，内容从 ``<tool_call>`` 本身开始返回，
        保证 tool parser 仍能收到结构完整的一次调用。"""
        ids = [THINK_S, TEXT, TOOL_S, 30, TOOL_E]
        assert parser.extract_content_ids(ids) == [TOOL_S, 30, TOOL_E]

```

```

def test_previous_turn_tool_call_ignored(self, parser):
    # 上一轮已闭合的 tool call 位于 <observation> 之前；回合边界守卫
    # 必须让它对当前轮「不可见」，输入原样返回。
    ids = [TOOL_S, 30, TOOL_E, OBSERVATION, TEXT, ASSISTANT, THINK_S, TEXT]
    assert parser.extract_content_ids(ids) == ids

def test_no_markers_returns_input_ids(self, parser):
    # 没有任何标记时不做裁剪。
    assert parser.extract_content_ids([20, 21]) == [20, 21]

def test_thinking_disabled(self, no_thinking_parser):
    # thinking=False 走基类 wait_for_reasoning 早返回路径，原样透传。
    ids = [THINK_S, TEXT, TOOL_S]
    assert no_thinking_parser.extract_content_ids(ids) == ids

```

评论区精华

review 中出现了一次典型的「作者初版实现有 blocking bug → 外部 (AI) 审查给出反例与 patch → 维护者要求集中到基类 → rebase 后收口」的完整闭环，讨论密度高 (22 条评论、3 条 review comment)。

- gaby (Claude Opus 5 复审) 指出 `extract_content_ids` 丢内容：用具体 id 序列说明反向扫描只返回最后一个 marker 之后的内容，导致 `</think>` 与 tool call 之间的 20, 21 被静默丢弃；并给出正确优先级「reasoning-end token anywhere in the sequence first, tool marker only as fallback」，同时附上可直接应用的 patch 与测试文件。
- JaredforReal 要求把修复放到基类：在 `vllm/parser/glm47_moe.py` 上评论「Can we fix this issue in the base ParserEngine instead of adding another override, which would be cleaner and easier for further maintenance」，并提醒 base 已有 `_derive_reasoning_end_token_ids`、`is_reasoning_end`，且 `ling3` 继承 GLM parser、需复检其 turn boundary。
- JaredforReal 报告基类改动打破 `qwen / dsv41` 测试：指出「seems like the current implementation break the qwen and dsv41's tests」，建议「we may need an opt-in path for other model」，同时表态「Moving into base is the right direction IMO」。作者解释失败来自 PR #56200 落地前硬编码的 `initial_state` 守卫，rebase 后严格遵循 `wait_for_reasoning`，4,314 个用例在 `qwen3`、`dsv41`、`deepseek v32/v4/v41`、`ling3`、`minimax`、`nemotron`、`glm` 上全部通过，因此无需 opt-in flag。
- chaunceyjiang 追问覆盖范围：只问一句「Have you tested any other models?」，作者给出上述测试矩阵作为答复。
- 作者主动撤回越界声明：从「dropped a fix for the token bleed」改为承认「dropping the 56605 claim and rewriting this to use the base reasoning_end_token_ids with proper tests」，并把 gaby 的 patch 直接应用到分支。
- 流程性讨论：gaby 提醒修 DCO (`git commit --amend -s && git push --force`)，`mergify[bot]` 报合并冲突并指出冲突由 #56200 合并引起，作者 rebase 后经 `/ci run` 触发 Buildkite CI #89284；`ywang96` 收尾时打趣「@JaredforReal is the real hero here!」。

- 反向扫描导致 `</think>` 与 tool call 之间的正文被静默丢弃 (correctness): 作者承认「the backward scan definitely trips over previous turns without the boundary guards and drops content on mixed deltas」, 直接应用 gaby 的 patch: 改为先反向定位回合起点、再正向扫描首个结束符, 并补齐 `test_think_end_wins_over_later_tool_call` 等断言。
- 修复应放在基类还是模型侧 override (design): 作者接受建议, 把多结束符 + 回合边界逻辑上移到 `ParserEngine.extract_content_ids`, 删除 `Glm47MoeParser.extract_content_ids` 与 config 中手写的 `wait_for_reasoning=thinking`, GLM 侧只保留 `turn_boundary_tokens=GLM_TURN_BOUNDARIES` 声明; ling3 等继承者因此自动受益。
- 基类改动对 qwen3 / dsv41 等其它 parser 的兼容性与是否需要 opt-in (testing): rebase 到 main 后 `extract_content_ids` 严格遵循 `wait_for_reasoning`, 作者本地跑通 `tests/parser/engine/` 共 4,314 个用例 (qwen3、dsv41、deepseek v32/v4/v41、ling3、minimax、nemotron、glm), 结论是不需要 opt-in flag; JaredforReal 随后给出「LGTM now」。
- 多轮 prompt 中上一轮 reasoning / tool 标记被误判为当前轮状态 (correctness): 最终未直接改 `ParserEngine.is_reasoning_end` (其 boundary 分支已存在于基类), 而是通过 GLM 新增 `GLM_TURN_BOUNDARIES` 注入 `_turn_boundary_token_ids`, 并用 `test_previous_turn_tool_call_ignored`、`test_previous_turn_think_end_ignored` 固化行为。
- DCO 签名与合并冲突处理 (other): 作者补签 DCO 并 rebase 到 main 解冲突, 随后 `/ci run` 触发 Buildkite CI #89284; PR 由 ywang96 合并, 作者随后删除分支。

风险与影响

• 风险:

1. 基类公共方法的回归面: `ParserEngine.extract_content_ids` 被所有 declarative parser 复用, 改动的第二步 (`wait_for_reasoning` 早返回、多结束符前向扫描) 对所有子类生效。作者以本地 4,314 个用例通过作为证据, 但 `tests/parser/engine/` 都是 mock tokenizer 的 token-id 单测, 缺少真实 chat template + OpenAI tool-calling 的端到端断言。
2. 未声明回合边界的多结束符 parser: `turn_start` 默认 0, 若某模型同时配置了多个 reasoning 结束符却没设 `turn_boundary_tokens`, 仍可能把历史轮的结束符当作当前轮, 导致 content 截断位置偏前。GLM 已修, 继承 GLM 的 ling3 只在 review 中被口头提醒, 本 PR 未加对应测试。
3. streaming 分块下的鲁棒性: 反向扫描依赖「最后一个 boundary token 在序列内」, 若 delta 从回合中间开始 (没有 boundary 出现), `turn_start` 会保持 0, 行为退化为「在可见窗口内取第一个结束符」, 属于保守回退而非崩溃, 但对首包即含 `</think>` 的极端分块需要额外确认。
4. `wait_for_reasoning=thinking` 被删除: 语义等价性依赖「initial_state 与 `wait_for_reasoning` 同源」这一隐含契约, 未来若有人只改 `initial_state` 而不知其推导

作用，可能产生难以察觉的行为漂移；建议在 ParserEngineConfig 上补注释或断言。

5. 无性能与兼容性风险：新增逻辑是单趟 $O(n)$ 扫描，无 API/schema 变更，不涉及分布式或部署配置。 - 影响：对用户：GLM-4.7（及继承其 parser 的 ling3 等）在 thinking → tool call 场景下不再丢正文、不再把上一轮标记算进当前轮，tool-calling 输出结构更完整；thinking 关闭的请求仍原样透传，无行为变化。对系统：改动落在 vllm/parser/engine/parser_engine.py 这一跨模型共享基类，影响 qwen3、dsv41、deepseek 系列、ling3、minimax、nemotron、glm 全部 declarative parser，是 parser frontend 的公共路径变更，需按核心路径对待。对团队：本 PR 确立了「边界 / 结束符一类语义放基类、模型侧只做声明式配置 (turn_boundary_tokens)」的维护范式，后续新增带隐式 reasoning 终止符的模型时应复用该模式，而不是再写 extract_content_ids override。 - 风险标记：核心路径变更，跨模型解析器兼容，依赖上游 wait_for_reasoning 语义，缺少端到端验证，边界 token 未声明时退化

关联脉络

- PR #56200（材料未给出标题）引入 parser wait_for_reasoning 的上游基类改动：讨论中被两次点名：JaredforReal 指出本 PR 上一版因此改动而失败，gaby 指出合并冲突由该 PR 引起；本 PR 最终实现也依赖 wait_for_reasoning（在 config 中删除显式赋值后由基类从 initial_state 推导）。
- PR #56605（材料未给出标题）GLM tool call 中的 token bleed 问题：作者最初声称本 PR 修复该 token bleed 问题并 cc 了正在看 #56605 的维护者，gaby 明确表示「I don't think it fixes #56605」，作者随后撤回该声明，把 PR 收敛为 parser frontend 的边界处理。
- PR #57045 [UX] Add thinking support to vllm chat: 同属 parser frontend 的 reasoning/thinking 呈现脉络：那条 PR 让 CLI 流式展示 reasoning 内容，本 PR 修正了 reasoning 结束与边界的 token 判定，二者共同决定 thinking 输出的正确切分。
- PR #56988 [Bugfix][Responses] Clean up MCP tool sessions once before closing them: 同属 tool-calling 前端链路：那一项修复 Responses 接口的 MCP 工具会话生命周期，本 PR 修复工具调用与 reasoning 边界的 token 识别，都是 tool-calling 请求端到端正确性的组成部分。