PR #34881 完整报告

sgl-project/sglang

Stop losing Kimi-K3 tool calls to reasoning, constraint conflicts, and truncation

合并时间：2026-08-19 03:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34881

执行摘要

- 一句话：修复 Kimi-K3 工具调用四处丢失缺陷，消除静默丢弃
- 推荐动作：值得精读。这份 PR 的根因分析方法论（先确认"报错方"与"吞错方"分离，再逐条追踪静默路径）有很高示范价值；detector 能力门控是干净可复用的设计。建议重点关注三处：_process_tool_calls 的门控与形状校验、finish() 的流末记账、protocol.py 的 400 决策——后者是典型的"宁可显式失败，也不静默降级"工程取舍，值得在 API 兼容性策略讨论中引用。

功能与动机

Issue #34604 报告 Kimi-K3 在 agentic 生产环境每天约 190 次 Tool call parsing error，且请求会退回纯文本，导致 agent 循环退化、每 7-8 分钟丢一次工具调用。PR 作者追踪后指出检测器并非根源——它使用 stdlib json 且已经吞噬所有异常——真正原因是 serving 层把原生格式输出推给 orjson 解码、required 约束被静默丢弃等四个独立缺陷，其中两个完全静默，因此 190/ 天只是下限而非全貌。PR body 明确写道："the detector isn't the source — it uses stdlib json... and detect_and_parse already swallows every exception. Tracing it produced four distinct defects; two of them are entirely silent, so 190/day is a floor, not the rate."

实现拆解

1. 重构 _process_tool_calls（serving_chat.py）：将原先的 should_try_parser 拆成 detector_owns_format = supports_structural_tag() or parses_required_natively()，使 should_try_parser = not is_required or detector_owns_format。这样 kimi_k3 这类自带结构标签约束的检测器在 tool_choice=required 时若解析不到调用，会直接 warn 并返回文本，不再落入 orjson.loads 的 JSON 数组解码器——因为输出根本不可能是 JSON 数组。同步给 JSON fallback 加了形状校验：裸 dict 射成单元素数组、缺失 parameters 默认 {}、非数组或元素缺 name 时抛出可读错误而非 TypeError。
2. 对齐 Responses API（serving_responses.py）：_make_response_output_items 沿用同样的能力门控，并补上 parses_required_natively()，让 muse（ATEM 格式）在 required 下也跳过 JSON fallback，避免 Chat 与 Responses 两条 API 行为不一致。
3. 协议层拒绝不可满足的组合（protocol.py）：to_sampling_params 检测到 tool_choice=required 或命名工具同时携带 response_format / regex / ebnf 时直接抛 ValueError（上抛为 400）。原因是 sglang 只有一个 grammar 槽位，该组合本质上无法

同时满足；auto 继续 warn-and-continue，因为 auto 不承诺必须调用。

4. 修复 reasoning 解析次序（reasoning_parser.py）：detect_and_parse 与 parse_streaming_increment 均改为比较 think_end 与 tool_start_token 谁先出现，tools 通道先出现时按 tools 分割，避免完整工具调用被吞入 reasoning_text 静默丢失。
5. 补齐流末上报（kimik3_detector.py）：新增 finish() 覆盖，检测 buffer 中残留的截断 tools 段并 warn；无 tools 标记时释放被 hold 的普通文本，并用 strip_partial_marker_suffix 清理半截 <|open|> 类标记。
6. 测试配套：4 个测试文件新增约 290 行用例，覆盖 native parser 跳过 fallback、截断 warning、fallback 形状容错、冲突 400、auto 不报错、流式多种 chunk 大小下 think/tools 次序，以及 finish() 三条路径。作者记录 pre-fix 为 8 failed / 6 passed，post-fix 为 0 failed。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 API 服务；类别 source；类型 core-logic；符号 _process_tool_calls）：主修复点：required 下结构标签解析器不再落入 orjson JSON fallback，fallback 形状校验加固，同步路径补截断 warning。
- python/sglang/srt/function_call/kimik3_detector.py（模块 工具调用；类别 source；类型 core-logic；符号 finish）：新增 finish() 覆盖，修复流式路径截断 tools 段零日志静默消失，并释放 held-back 文本。
- python/sglang/srt/parser/reasoning_parser.py（模块 推理解析；类别 source；类型 core-logic；符号 detect_and_parse, parse_streaming_increment）：修复 kimi_k3 覆盖分支中 think_close/tool_start 次序问题，同步与流式路径都避免工具调用被吞入 reasoning。
- python/sglang/srt/entrypoints/openai/serving_responses.py（模块 API 服务；类别 source；类型 core-logic；符号 _make_response_output_items）：与 Chat API 对齐 required 门控，补充 parses_required_natively()，避免 muse 等原生格式检测器被 orjson fallback 误伤。
- python/sglang/srt/entrypoints/openai/protocol.py（模块 请求协议；类别 source；类型 core-logic；符号 to_sampling_params）：required/named 与 response_format/regex/ebnf 冲突时直接 400，避免约束被静默丢弃后产生不可满足请求。
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_required_tool_choice_skips_json_fallback_for_native_parser, test_truncated_native_tool_call_logs_and_drops, test_required_tool_choice_json_fallback_tolerates_odd_shapes, test_required_tool_choice_rejects_conflicting_output_constraint）：5 个新用例覆盖 native parser 跳过 fallback、截断 warning、fallback 形状容错、冲突 400、auto 不报错，是行为变更的主要回归网。
- test/registered/function_call/test_kimik3_detector.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_stream_end_reports_truncated_tools_section, test_stream_end_releases_held_back_text, test_stream_end_drops_truncated_marker）：覆盖 finish() 三条路径：截断 tools 上报、held-back 文本释放、半截标记丢弃。

- test/registered/unit/parser/test_kimik3_reasoning_parser.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_non_stream_tools_channel_before_think_close_is_not_reasoning, test_streaming_tools_channel_before_think_close）：验证 tools 通道先于 think 关闭时不被吞入 reasoning，流式覆盖 4 种 chunk 大小。
- test/registered/unit/entrypoints/openai/test_serving_responses.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_required_tool_choice_skips_json_fallback_for_native_parser）：确保 muse（parses_required_natively）在 Responses API 路径同样跳过 JSON fallback，防止两条 API 行为分叉。

关键符号：_process_tool_calls, _make_response_output_items, to_sampling_params, finish, detect_and_parse, parse_streaming_increment

关键源码片段

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

主修复点：required 下结构标签解析器不再落入 orjson JSON fallback，fallback 形状校验加固，同步路径补截断 warning。

```
# 关键变更 1：用 detector 能力门控替代旧的 should_try_parser。
# detector_owns_format 表示该检测器自己约束生成格式（structural_tag 或原生格式），
# 此时 required 下解析不到调用就不应落入 orjson JSON fallback ——
# 输出根本不会是 JSON 数组，强行解码只会得到晦涩的 TypeError。
detector_owns_format = (
    parser.detector.supports_structural_tag()
    or parser.detector.parses_required_natively()
)
should_try_parser = not is_required or detector_owns_format

if should_try_parser and parser.has_tool_call(text):
    try:
        text, call_info_list = parser.parse_non_stream(text)
        if not call_info_list:
            # 修复 4（同步路径）：tools 标记存在但零完整调用时，
            # 过去直接 return 且无任何日志；现在明确 warn 并附 debug 原文。
            logger.warning(
                "Tool call marker present but no complete call parsed "
                "from %s output; dropping the incomplete call",
                self.tool_call_parser,
            )
            logger.debug(
                "Unparsed tool call output (%d chars): %r", len(text), text[:2000]
            )
            return ToolCallProcessingResult(None, text, finish_reason)
        # ... 正常组装 ToolCall 列表，省略 ...
    except Exception as e:
        logger.error(f"Tool call parsing error: {e}")
        return ToolCallProcessingResult(None, text, finish_reason)
```

```
# 修复 1：required 且 detector 拥有格式但没产出调用时直接返回文本，
# 绝不再走下面的 orjson JSON 数组解码。
if is_required and detector_owns_format:
    logger.warning(
        "Required tool call missing from %s output (%d chars)",
        self.tool_call_parser,
        len(text),
    )
    logger.debug("Unparsed required tool call output: %r", text[:2000])
    return ToolCallProcessingResult(None, text, finish_reason)

# 修复 1 的另一半：真正使用 JSON fallback 的解析器（如 glm45）
# 加固形状校验，把裸 dict、缺 parameters、非数组统一收敛为可读错误。
try:
    tool_call_data = orjson.loads(text)
    if isinstance(tool_call_data, dict):
        tool_call_data = [tool_call_data]
    if not isinstance(tool_call_data, list):
        raise ValueError(
            "expected a JSON array of tool calls, got "
            f"{type(tool_call_data).__name__}"
        )
    if not all(isinstance(tool, dict) and "name" in tool for tool in tool_call_data):
        raise ValueError("every tool call must be a JSON object with a 'name'")
    # ... 随后用 tool.get("parameters", {}) 兜底缺失参数 ...
```

python/sglang/srt/function_call/kimik3_detector.py

新增 finish() 覆盖，修复流式路径截断 tools 段零日志静默消失，并释放 held-back 文本。

```
# 修复 4（流式路径）：KimiK3Detector 原本不覆盖 finish()，
# 流结束时截断的 tools 段既无调用、无文本、也无日志，完全静默丢失。
def finish(self, tools: List[Tool]) -> StreamingParseResult:
    open_idx = self._buffer.find(self.bot_token)
    if open_idx != -1:
        # buffer 里还残留 tools 开头标记，说明生成被截断在 tools 段内。
        section = self._buffer[open_idx + len(self.bot_token):]
        if not self._parse_calls(section):
            logger.warning(
                "Kimi K3 tools section ended with no complete tool call; "
                "dropping %d buffered chars",
                len(section),
            )
        # 截断的工具调用不可恢复，丢弃即可，关键是给出日志。
        return StreamingParseResult()
    # 没有 tools 开头标记：释放之前因后缀歧义而被 hold 住的普通文本，
    # 并剥掉可能残留的半截 <|openl|> 等标记。
    pending = self._emit_normal_text(limit=len(self._buffer))
    return StreamingParseResult(normal_text=strip_partial_marker_suffix(pending))
```

python/sglang/srt/parser/reasoning_parser.py

修复 kimi_k3 覆盖分支中 think_close/tool_start 次序问题，同步与流式路径都避免工具调用被吞入 reasoning。

```
# 修复 3：tools 通道先于 think 关闭时不应被算作 reasoning。
# 此前只在 think_end 缺失时才做 tools 通道救援；一旦 think_close 存在，
# 位于它之前的完整 tools 段会整段落入 reasoning_text，工具调用被静默吞掉。
def detect_and_parse(self, text: str) -> StreamingParseResult:
    open_idx = text.find(self.think_start_token)
    start = open_idx + len(self.think_start_token) if open_idx != -1 else 0
    close_idx = text.find(self.think_end_token, start)
    tools_idx = text.find(self.tool_start_token, start)
    # 取两者中更早出现的作为分割点：tools 先出现说明模型在思考结束前就开了工具通道。
    if close_idx != -1 and tools_idx != -1 and tools_idx < close_idx:
        return StreamingParseResult(
            reasoning_text=strip_partial_marker_suffix(text[start:tools_idx]),
            normal_text=self._clean_content(text[tools_idx:]),
        )
    # ... 其余原逻辑 ...

# 流式增量路径使用同样的次序判断，避免分块到达时把 tools 段误判进 reasoning。
def parse_streaming_increment(self, new_text: str) -> StreamingParseResult:
    close_idx = buf.find(self.think_end_token)
    tools_idx = buf.find(self.tool_start_token)
    if close_idx != -1 and not (tools_idx != -1 and tools_idx < close_idx):
        reasoning_text = buf[:close_idx]
        self._buffer = buf[close_idx + len(self.think_end_token):]
        self._in_reasoning = False
```

评论区精华

评审只有一条 APPROVED，但 PR body 内包含大量设计取舍说明：

- required + response_format 改为 400 的兼容性权衡：审核者 JustinTong0323 在批准时特别提醒："The required + response_format 400 is called out in the body — heads-up that it is stricter than OpenAI and worth a maintainer nod before merge." 即 OpenAI 会容忍该组合尽力而为，sglang 选择直接拒绝，属于更严格的 API 契约。
- required 仍不是硬保证：PR body 自述 EOS 已被 grammar 屏蔽，但 max_tokens 和客户端 stop 序列可以在解码文本上直接终止、不经过 grammar（schedule_batch.py 的 _check_str_based_finish），作者明确留待单独 issue，不在本 PR 处理。
- 同源 bug 的取舍：作者发现同一 think_end/tool_start 顺序 bug 存在于 BaseReasoningFormatDetector 下另外 6 个设置 tool_start_token 的检测器，但只在 kimi_k3 上实证过，刻意不动以免一次性回归 6 个模型家族。
  - required + response_format 改为 400 的兼容性权衡 (design): 维持 400，PR 已被批准合并；sglang 单一 grammar 槽位使该组合本质上不可满足，显式失败优于静默降级。

- required 仍非硬保证：max_tokens 与 stop 会绕过 grammar 屏蔽 (question): 未在本 PR 解决，作者明确留待后续 issue 跟踪。
- BaseReasoningFormatDetector 中同源顺序 bug 是否顺手修复 (design): 本次不修，仅修复 kimi_k3 覆盖分支，留待后续按模型逐个验证。

风险与影响

- 风险：
  - API 行为变更（breaking change）：protocol.py 对 required/ 命名工具 + response_format/regex/ebnf 的组合从“警告后继续但必失败”改为直接 400，比 OpenAI 更严格，存量依赖该组合的客户端会立刻报错，需要发布说明与升级指引。
  - 核心请求路径变更：serving_chat.py 的 _process_tool_calls 与 serving_responses.py 的 _make_response_output_items 是每次工具调用请求的必经之路，门控逻辑改写存在影响其它检测器（如 glm45 仍走 JSON fallback）的回归风险；不过形状校验只会把原本的 TypeError 变成可读错误并回退文本，方向是安全的。
  - 流式状态机记账改动：kimik3_detector.py 新增的 finish() 涉及 _buffer / _sent_normal_idx 记账，若 off-by-one 可能导致文本重复或丢失；测试覆盖了 held-back 释放与半截标记丢弃两条路径，风险可控。
  - 推理解析基类路径：reasoning_parser.py 的改动只命中 kimi_k3 覆盖分支，未改 BaseReasoningFormatDetector 本身，但该基类仍保留同源顺序 bug，后续如果其它模型也暴露此问题，需要按相同模式跟进。
- 影响：
  - 用户侧：Kimi-K3 agentic 用户不再每 7-8 分钟丢一次工具调用，190/ 天的解析错误日志基本消除；截断的工具调用从“零日志静默消失”变为可观测的 warning，运维定位成本大幅下降。
  - API 契约：required + response_format 组合从“必失败但延迟暴露”变为“请求时即 400”，是明确的行为收紧，需同步文档与 release note。
  - 团队协作：为 servoing 层确立“detector 是否拥有输出格式”的统一能力门控（supports_structural_tag / parses_required_natively），后续接入新检测器时必须在两种能力上显式声明，否则 required 路径的 fallback 行为会不一致。
  - 风险标记：API 行为变更（400 拒绝），核心请求路径变更，流式状态机记账改动，跨模块行为对齐（chat/responses）

关联脉络

- PR #34627 fix: preserve output logprobs without input logprobs: 同为 serving/entrypoints 层的静默行为修复，都强调测试先行与 API 契约的显式化，反映同一条维护主线。
- PR #34604 [Bug] Kimi-K3 tool call parser fails ~8x/hour in production: 本 PR 的直接驱动 Issue，报告中 190/day 的解析错误正是本 PR 修复的四种缺陷的显式表现。