PR #34458 完整报告

sgl-project/sglang

[Fix] Make DeepSeek-V4 reasoning and tool-call streaming parsing chunk-invariant

合并时间：2026-08-12 11:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34458

执行摘要

- 一句话：DSV4 流式推理与工具调用解析改为 chunk 无关
- 推荐动作：值得精读。核心看点是 holdback 设计（最长前缀匹配 + finish 冲刷）与 TestStreamingChunkSizeInvariance 的多 chunk size 不变量测试，这两种手法对任何做增量解析的流式系统都有借鉴意义；同时建议关注 PR body 中显式固定的两个已知分歧，后续涉及 DeepSeek-V4 解析的改动必须同步更新对应测试。

功能与动机

PR body 指出：推测解码和 stream_interval > 1 每步交付多个 token，把多字符标记拆到 chunk 边界；_parse_streaming_increment_impl 每次发射后清空 _buffer 并破坏碎片，导致结束标记从未被识别，turn 剩余部分泄漏进 reasoning_content。作者给出具体反例：main 上 'abcnormal' 在 chunk size 3 下产出 reasoning='abcnormal'、normal=''，且该问题影响所有共享基类实现的检测器，不止 DSV4。PR 还声明本修复替代了 #31009、#21179、#32539、#32332、#28684、#31786、#33813、#34244、#23786、#34157、#34178 等一批散落的补丁。

实现拆解

1. 推理解析器引入 holdback 机制：在 python/sglang/srt/parser/reasoning_parser.py 的 BaseReasoningFormatDetector._parse_streaming_increment_impl 中，stream_reasoning=True 时不再清空 _buffer，而是用 _ends_with_partial_token 找出最长“是 think_end_token / tool_start_token / think_start_token 严格前缀”的尾随切片扣在 _buffer 里，其余内容作为 reasoning 发射；剥离 后把结果写回 _buffer，避免 stream_reasoning=False 时 finish() 把原始标签带进 reasoning_content。
2. 工具块路由：DeepSeekV4Detector.__init__传入 tool_start_token="<｜DSML｜"（含前导 <，与 has_tool_call 匹配一致），使无 的 DSML 块从 reasoning_content 转入 normal_text 交给工具调用检测器。
3. 工具调用检测器（python/sglang/srt/function_call/deepseekv32_detector.py）：捕获 MalformedJSON 回退为原始值；detect_and_parse 用 findall 扫描所有 tool_calls 段；parse_streaming_increment 在首个 invoke 前恢复 preamble 并做与 detect_and_parse 相同的尾部换行修剪；解析异常时丢弃 buffer 原样重发文本，并丢弃半成品调用避免无参调用到达客户端。

4. 测试配套：TestStreamingChunkSizeInvariance 用 [1, 2, 3, 5, 7, 11, 23, 1000] 断言流式结果与 chunk 划分无关且与 detect_and_parse 一致，并显式固定两个已知 chunk 依赖分歧；新增 test/registered/unit/function_call/test_deepseekv4_detector.py 覆盖 preamble、MalformedJSON、错误恢复与多段解析；翻转 test_finish_drops_partial_end_tag… 与 GLM-4.5 stream_reasoning=False 断言以匹配新语义。

关键文件：

- test/registered/unit/function_call/test_deepseekv4_detector.py（模块 工具调用测试；类别 test；类型 test-coverage；符号 _wrapped, _invoke, _param, _weather_call）：新增的 DSV4 DSML 流式单元测试，不依赖服务器与模型加载，覆盖 preamble 保留、MalformedJSON 回退、错误路径恢复与多 tool_calls 段解析，是工具调用检测器改动的直接回归保障。
- test/registered/unit/parser/test_reasoning_parser.py（模块 解析器测试；类别 test；类型 test-coverage；符号 test_finish_drops_partial_end_tag_when_streaming_reasoning, test_finish_flushes_partial_end_tag_when_streaming_reasoning, test_dsml_block_is_routed_out_of_reasoning, TestStreamingChunkSizeInvariance）：核心回归测试载体：新增 TestStreamingChunkSizeInvariance 用 8 种 chunk size 验证解析结果与 chunk 划分无关，并显式固定两个已知 chunk 依赖分歧；同时翻转两个旧断言以匹配 finish() 与 GLM-4.5 的新语义。
- python/sglang/srt/parser/reasoning_parser.py（模块 推理解析器；类别 source；类型 core-logic；符号 _ends_with_partial_token）：本次修复的核心：BaseReasoningFormatDetector 增加 holdback 机制、_ends_with_partial_token 上移基类并改为最长匹配、finish() 从丢弃改为冲刷，同时 DeepSeekV4Detector 传入 tool_start_token 使 DSML 块路由出 reasoning。
- python/sglang/srt/function_call/deepseekv32_detector.py（模块 工具调用；类别 source；类型 core-logic；符号 parse_streaming_increment, detect_and_parse, _parse_parameters_from_xml）：DSML 工具调用检测器的行为修正：MalformedJSON 不再逃逸、detect_and_parse 解析全部 tool_calls 段、流式解析保留 preamble 并在错误路径丢弃 buffer 重新发射文本。

关键符号：BaseReasoningFormatDetector._parse_streaming_increment_impl, BaseReasoningFormatDetector._ends_with_partial_token, BaseReasoningFormatDetector.finish, DeepSeekV4Detector.init, DeepSeekV32Detector.parse_streaming_increment, DeepSeekV32Detector.detect_and_parse, DeepSeekV32Detector._parse_parameters_from_xml, TestStreamingChunkSizeInvariance, TestDeepSeekV4Streaming

关键源码片段

test/registered/unit/parser/test_reasoning_parser.py

核心回归测试载体：新增 TestStreamingChunkSizeInvariance 用 8 种 chunk size 验证解析结果与 chunk 划分无关，并显式固定两个已知 chunk 依赖分歧；同时翻转两个旧断言以匹配 finish() 与 GLM-4.5 的新语义。

```
class TestStreamingChunkSizeInvariance(CustomTestCase):
    """累计的 (reasoning, normal) 输出不应依赖 decode 步骤如何批量 token,
    并且必须与一次性 detect_and_parse 一致。

    推测解码与 stream_interval > 1 每次交付多个 token,
    会把 </think> 这类多字符标记拆到 chunk 边界。
    两个 _is_chunk_dependent 测试固定了已知偏差。
    """

    CHUNK_SIZES = [1, 2, 3, 5, 7, 11, 23, 1000]
    DSML = "｜DSML｜"

    def _feed(self, detector, text, chunk_size):
        reasoning = normal = ""
        for i in range(0, len(text), chunk_size):
            result = detector.parse_streaming_increment(text[i:i + chunk_size])
            reasoning += result.reasoning_text
            normal += result.normal_text
        # finish() 冲刷被扣住的 "｜半截标记｜", 也必须计入结果
        result = detector.finish()
        return reasoning + result.reasoning_text, normal + result.normal_text

    def _assert_invariant(self, make_detector, text, expected):
        for chunk_size in self.CHUNK_SIZES:
            with self.subTest(chunk_size=chunk_size):
                self.assertEqual(
                    self._feed(make_detector(), text, chunk_size), expected)
        # 流式累计结果必须与非流式单次解析一致
        one_shot = make_detector().detect_and_parse(text)
        self.assertEqual((one_shot.reasoning_text, one_shot.normal_text), expected)

    def test_think_end_split_across_chunks(self):
        # </think> 正好跨越 chunk 边界也必须结束推理块
        self._assert_invariant(
            DeepSeekR1Detector,
            "<think>abc reasoning</think>normal text",
            ("abc reasoning", "normal text"),
        )

    def test_reasoning_truncated_mid_partial_token(self):
        # 推理恰好以 </think> 前缀结尾时这些字符必须保留:
        # 扣住是为了与下一块拼回标记, 若流先结束则应冲刷而不是吞掉
        for chunk_size in self.CHUNK_SIZES:
            with self.subTest(chunk_size=chunk_size):
                self.assertEqual(
                    self._feed(DeepSeekR1Detector(), "<think>compare a <", chunk_size),
                    ("compare a <", ""),
                )
```

python/sglang/srt/function_call/deepseekv32_detector.py

DSML 工具调用检测器的行为修正：MalformedJSON 不再逃逸、detect_and_parse 解析全部 tool_calls 段、流式解析保留 preamble 并在错误路径丢弃 buffer 重新发射文本。

```
  def parse_streaming_increment(self, new_text: str,
                                tools: list[Tool]) -> StreamingParseResult:
      self._buffer += new_text
      current_text = self._buffer

      # 没有 DSML 痕迹时直接透出文本；保留可能成为标签的尾缀
      dsml_markers = ["｜DSML｜", "<｜", "</｜"]
      potentially_dsml = any(marker in current_text for marker in dsml_markers)
      dsml_prefixes = ["<", "<｜", "</", "</｜"]
      ends_with_prefix = any(current_text.rstrip().endswith(p)
                             for p in dsml_prefixes)

      if (not self.has_tool_call(current_text)
              and not potentially_dsml and not ends_with_prefix):
          self._buffer = ""
          for e_token in [self.eot_token, self.invoke_end_token]:
              if e_token in current_text:
                  current_text = current_text.replace(e_token, "")
          return StreamingParseResult(normal_text=current_text)

      all_calls: list[ToolCallItem] = []
      # 只对第一个 invoke 恢复前置正文：DSML 守卫不会释放仍含标记的 buffer，
      # 所以后续 prose 会留在缓冲里；这里与 detect_and_parse 行为对齐
      preamble = ""
      try:
          while True:
              invoke_match = re.search(self.invoke_regex, current_text, re.DOTALL)
              if not invoke_match:
                  break

              func_name, invoke_content, is_tool_end = \
                  self._unpack_invoke_match(invoke_match)

              if self.current_tool_id == -1:
                  self.current_tool_id = 0
                  self.prev_tool_call_arr = []
                  self.streamed_args_for_tool = [""]
                  # 恢复工具调用之前的正文（含 bot_token 定位与尾部换行修剪）
                  call_start = invoke_match.start()
                  bot_pos = current_text.rfind(self.bot_token, 0, call_start)
                  if bot_pos != -1:
                      call_start = bot_pos
                  # 与 detect_and_parse 相同的尾部换行修剪，保证两者一致
                  preamble = current_text[:call_start].removesuffix("
```

```
")

            while len(self.prev_tool_call_arr) <= self.current_tool_id:
                self.prev_tool_call_arr.append({})
            while len(self.streamed_args_for_tool) <= self.current_tool_id:
                self.streamed_args_for_tool.append("")

            if not self.current_tool_name_sent:
                all_calls.append(ToolCallItem(
                    tool_index=self.current_tool_id,
                    name=func_name, parameters=""))
                self.current_tool_name_sent = True

            current_params = self._parse_parameters_from_xml(
                invoke_content, allow_partial=not is_tool_end)

            sent_len = len(self.streamed_args_for_tool[self.current_tool_id])
            prev_params = self.prev_tool_call_arr[self.current_tool_id].get(
                "arguments")
            argument_diff = None
            if prev_params is None or current_params != prev_params:
                argument_diff = current_params[sent_len:]
                if argument_diff:
                    all_calls.append(ToolCallItem(
                        tool_index=self.current_tool_id, name=None,
                        parameters=argument_diff))
                    self.streamed_args_for_tool[self.current_tool_id] += \
                        argument_diff

            self.prev_tool_call_arr[self.current_tool_id] = {
                "name": func_name, "arguments": current_params}

            if is_tool_end:
                self._buffer = current_text[invoke_match.end():]
                current_text = self._buffer
                self.current_tool_id += 1
                self.current_tool_name_sent = False
                continue
            break

        return StreamingParseResult(normal_text=preamble, calls=all_calls)

    except Exception as e:
        logger.error(f"Error in parse_streaming_increment: {e}")
        # 出错时丢掉 buffer 并原样重发，避免每个后续 chunk 反复解析残局；
        # 丢弃半成品调用：失败可能落在工具名与参数之间，比发给客户端更好
        self._buffer = ""
        if not current_text.startswith(preamble):
            current_text = preamble + current_text
```

```
        return StreamingParseResult(normal_text=current_text)
```

评论区精华

该 PR 的 review 评论为空，issue 评论仅包含 CI 重跑指令。设计权衡主要体现在 PR body 的 Known divergences 与提交历史中：作者明确接受两个 chunk 依赖分歧（模型在推理中引用 DSML 标记、 前文本），并要求用 *_is_chunk_dependent 测试固定，未来修复必须同步更新测试而非静默改变行为。提交历史也展示了语义迭代过程，例如从 "keep finish() dropping unfinished end token" 到 "flush held-back suffix on finish; longest-match holdback; drop half-formed call on error"，再到 "revert multi-think-block handling; complete holdback with think_start"，反映了 holdback 范围与 finish() 语义的反复权衡。

- PR 无 review 评论，设计权衡记录在 body Known divergences 与提交历史 (design): 接受这两个已知分歧，要求后续修复必须同步更新测试，而不是静默改变行为。

风险与影响

- 风险：
  1. 核心路径变更：BaseReasoningFormatDetector 是每个推理 token 都要走的解析路径，本次改动影响所有共享基类的检测器（GLM-4.5、Qwen3、DeepSeek-R1、Inkling 等），回归面较广，但配套测试覆盖了主要模型。
  2. finish() 语义翻转：stream_reasoning=True 时 finish() 从 " 丢弃残留 fragment" 变为 " 冲刷被扣内容 "，推理以 < 等标记前缀结尾的场景会多输出内容，这是有意修复，但依赖旧行为的客户端需关注。
  3. 已知 chunk 依赖分歧：流式与一次性解析在 " 推理中引用 DSML 标记 " 与 " 前文本 " 两种场景下结果不同，属于接受的缺陷，可能造成用户困惑。
  4. 异常路径行为变化：deepseekv32_detector.py 解析失败时丢弃 buffer 并丢弃半成品调用，避免脏数据进入客户端，但也意味着失败时工具调用静默丢失。
  5. _ends_with_partial_token 每次推理发射做最长后缀匹配，复杂度 O(buffer 长度 × token 长度)，但 buffer 通常很小，性能风险低。 - 影响：对用户：修复了流式输出中 reasoning_content 混入 及后续正文、DSML 工具调用块滞留推理区、带前导 prose 的工具调用被吞、MalformedJSON 异常逃逸等行为，使流式与一次性解析结果对齐，是内容正确性的实质性提升。对系统：改动集中在解析层，不涉及调度与 kernel，风险可控，但所有推理模型共享该基类，影响面广。对团队：一次性合并替代了 11 个历史修复 PR，减少了分散补丁的维护负担，并建立了 chunk 不变性回归基线。 - 风险标记：核心路径变更（每 token 解析），finish() 语义翻转，已知 chunk 依赖分歧，共享基类影响多模型，异常路径行为变化

关联脉络

- PR #34262 [Feature] Add Muse Glimmer model support: 该 PR 同样修改了 python/sglang/srt/parser/reasoning_parser.py，本 PR 对 BaseReasoningFormatDetector 的 holdback 与 finish() 语义调整会影响其新增模型的推理解析路径。此外，PR body 声明本 PR 替代了 #31009、#21179、#32539、#32332、

#28684、#31786、#33813、#34244、#23786、#34157、#34178 等历史修复。