PR #45104 完整报告

vllm-project/vllm

[Refactor] Chat Completions Streaming Harmony Refactor and Bugfixes

合并时间：2026-06-12 09:09

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45104

执行摘要

- 一句话：重构流式路径统一到 HarmonyParser，删除专用流式模块
- 推荐动作：值得精读。展示了如何通过统一接口消除重复代码和修复顽固 bug。设计决策（将流式 delta 提取集成到 parser 内部）值得借鉴。

功能与动机

重构 Chat Completions Harmony 路径以使用统一的 Parser。此前 Harmony 绕过了统一 Parser 路径，非流式通过 harmony_utils.py 和 openai_tool_parser.py 流向，流式通过 stream_harmony.py 流向，导致双重解析和流式 bug（关联 issue #37070：流式工具调用崩溃和参数分裂）。本 PR 更新了统一 Parser 接口以接受 token ID，创建 HarmonyParser 适配 StreamableParser，并删除 Harmony 专用的流式服务路径，使两种模式都流经 parser.parse()/parse_delta()。

实现拆解

1. 删除旧流式模块：移除 vllm/entrypoints/openai/chat_completion/stream_harmony.py 及其测试 tests/entrypoints/openai/chat_completion/test_serving_chat_stream_harmony.py，其中的 TokenState 和 extract_harmony_streaming_delta 函数不再使用。
2. 增强 HarmonyParser：在 vllm/parser/harmony.py 中，HarmonyParser 类新增 parse_delta() 方法实现，该方法通过 process_chunk() 处理 token ID 序列，遍历生成的 Segment 列表并按照通道（channel）和接收者（recipient）分类组装 DeltaMessage。同时添加了 _next_tool_call_index 和 _num_processed_messages 状态跟踪，用于正确分配工具调用索引。删除了原有的 messages 属性（不再需要暴露原始消息列表）。
3. 统一服务调用路径：在 vllm/entrypoints/openai/chat_completion/serving.py 中，移除 use_harmony 标志和对应的 Harmony 专用分支代码，流式生成时不再额外创建 StreamableParser，而是直接使用 parser.parse_delta() 处理每个输出 token 序列。非流式路径保持不变（仍走 parser.parse()）。
4. 更新测试覆盖：在 tests/parser/test_harmony.py 中新增 TestParseDelta 测试类，覆盖基本流式场景、多token场景、工具调用跨delta分割场景等，确保parse_delta行为正确。删除了原有的 visible_segments 辅助函数，替换为 tool_call_headers、tool_call_payloads、combined_tool_arguments 等更贴合 delta 消息检验的辅助函数。

关键文件：

- vllm/entrypoints/openai/chat_completion/stream_harmony.py（模块 流式模块；类别 source；类型 deletion；符号 TokenState, extract_harmony_streaming_delta）：被删除的流式 Harmony 模块，其中包含 TokenState 和 extract_harmony_streaming_delta 函数，是本 PR 重构的核心移除对象。
- tests/entrypoints/openai/chat_completion/test_serving_chat_stream_harmony.py（模块 测试；类别 test；类型 deletion；符号 MockMessage, MockStreamableParser, TestExtractHarmonyStreamingDelta, test_final_channel_returns_content_delta）：被删除的流式模块测试，包含 TestExtractHarmonyStreamingDelta 等测试类。
- vllm/parser/harmony.py（模块 解析器；类别 source；类型 dependency-wiring；符号 messages）：核心解析器文件，新增了 parse_delta() 方法实现流式 delta 提取，并移除了 messages 属性。
- vllm/entrypoints/openai/chat_completion/serving.py（模块 服务层；类别 source；类型 dependency-wiring）：服务入口文件，移除了 use_harmony 标志和对应的 Harmony 专用流式代码路径，统一调用 parser.parse_delta()。
- tests/parser/test_harmony.py（模块 测试；类别 test；类型 test-coverage；符号 visible_segments, tool_call_tuples, tool_call_headers, tool_call_payloads）：测试文件，新增 TestParseDelta 测试类，覆盖流式场景。同时新增辅助函数 tool_call_headers、tool_call_payloads、combined_tool_arguments 用于验证 delta 消息。

关键符号：HarmonyParser.parse_delta, HarmonyParser.process_chunk, HarmonyParser.parse, HarmonyParser.init, extract_harmony_streaming_delta (deleted), TokenState (deleted)

关键源码片段

vllm/parser/harmony.py

核心解析器文件，新增了 parse_delta() 方法实现流式 delta 提取，并移除了 messages 属性。

```
    def parse_delta(
        self,
        delta_text: str,
        delta_token_ids: list[int],
        request: ChatCompletionRequest | ResponsesRequest,
        prompt_token_ids: list[int] | None = None,
        *,
        finished: bool,
    ) -> DeltaMessage | None:
        # 记录当前 recipient，用于检测工具调用边界
        prev_recipient = self.current_recipient
        # 通过 process_chunk 将 token ID 序列解析为 Segment 列表
        result = self.process_chunk(delta_token_ids)
        combined_content = ""
        combined_reasoning = ""
        tool_messages: list[DeltaToolCall] = []

        for segment in result.segments:
```

```
    # 跳过已完成的消息边界（boundary 段）
    if segment.completed_message is not None:
        prev_recipient = None
        continue

    # 判断段类型：推理、内容或工具调用
    segment_type = _SegmentType.from_channel_and_recipient(
        segment.channel, segment.recipient
    )
    match segment_type:
        case _SegmentType.REASONING:
            combined_reasoning += segment.delta
        case _SegmentType.CONTENT:
            combined_content += segment.delta
        case _SegmentType.TOOL:
            assert segment.recipient is not None
            if prev_recipient != segment.recipient:
                # 新工具调用开始：生成工具调用头部（含名称）
                tool_name = extract_function_from_recipient(segment.recipient)
                tool_messages.append(
                    DeltaToolCall(
                        id=make_tool_call_id(),
                        type="function",
                        function=DeltaFunctionCall(
                            name=tool_name,
                            arguments=segment.delta,
                        ),
                        index=self._next_tool_call_index,
                    )
                )
                self._next_tool_call_index += 1
                prev_recipient = segment.recipient
            elif segment.delta:
                # 已有工具调用的参数流：仅传递 arguments
                tool_call_index = self._next_tool_call_index - 1
                tool_messages.append(
                    DeltaToolCall(
                        index=tool_call_index,
                        function=DeltaFunctionCall(arguments=segment.delta),
                    )
                )

# 无任何内容则返回 None
if not combined_content and not combined_reasoning and not tool_messages:
    return None

# 组装最终 DeltaMessage
delta_message = DeltaMessage()
if combined_content:
```

```
            delta_message.content = combined_content
        if combined_reasoning:
            delta_message.reasoning = combined_reasoning
        if tool_messages:
            delta_message.tool_calls = tool_messages
        return delta_message
```

tests/parser/test_harmony.py

测试文件，新增 TestParseDelta 测试类，覆盖流式场景。同时新增辅助函数 tool_call_headers、tool_call_payloads、combined_tool_arguments 用于验证 delta 消息。

```
class TestParseDelta:
    """
    测试 HarmonyParser.parse_delta 的流式 delta 提取逻辑。
    每个测试使用 gpt_oss_tokenizer 构造真实的 token ID 序列。
    """

    def test_basic(self, gpt_oss_tokenizer, chat_request):
        """基本流式场景：先分析通道，再最终通道。"""
        parser = HarmonyParser(gpt_oss_tokenizer)
        # 第一个 token 序列：分析通道的推理内容
        first_delta = parser.parse_delta(
            delta_text="",
            delta_token_ids=encode_output("<|channel|>analysis<|message|>Thinking"),
            request=chat_request,
            finished=False,
        )
        # 第二个 token 序列：最终通道的回答内容
        second_delta = parser.parse_delta(
            delta_text="",
            delta_token_ids=encode_output(
                "<|end|><|start|>assistant<|channel|>final<|message|>Answer"
            ),
            request=chat_request,
            finished=False,
        )
        assert first_delta is not None
        assert first_delta.reasoning == "Thinking"
        assert first_delta.content is None
        assert second_delta is not None
        assert second_delta.content == "Answer"
        assert second_delta.reasoning is None

    def test_multi_token(self, gpt_oss_tokenizer, chat_request):
        """单次调用包含多个 token 的正确组装。"""
        parser = HarmonyParser(gpt_oss_tokenizer)
        delta = parser.parse_delta(
            delta_text="",
            delta_token_ids=encode_output("<|channel|>final<|message|>Hello, world!"),
```

```
            request=chat_request,
            finished=False,
        )
        assert delta is not None
        assert delta.content == "Hello, world!"
        assert delta.reasoning is None
        assert not delta.tool_calls

    @pytest.mark.parametrize("tool_channel", ["commentary", "analysis"])
    def test_tool_call_split_across_deltas(
        self, gpt_oss_tokenizer, chat_request, tool_channel
    ):
        """工具调用参数分布在多个 delta 中时的合并行为。"""
        parser = HarmonyParser(gpt_oss_tokenizer)
        # 第一次 delta：开始工具调用
        first_delta = parser.parse_delta(
            delta_text="",
            delta_token_ids=encode_output(
                f"<|channel|>{tool_channel}<|message|>"
            ),
            request=chat_request,
            finished=False,
        )
        # ... 后续 delta 包含参数
        # 此测试验证工具调用的 index 和 arguments 跨 delta 连续
```

评论区精华

无实质 review 评论，作者 yzong-rh 在 PR 评论中解释将非流式重构提取为 #45171，本 PR 专做流式重构。最终得到 sfeng33 的 LGTM 批准。

- 拆分非流式与流式重构为两个 PR (design): 接受拆分，两个 PR 先后合并。

风险与影响

- 风险：风险较低。原流式路径被完全替换，新路径通过 BFCL 和单元测试验证无回归。但若 harmony 模型（gpt-oss）有非预期行为，可能需要快速修复。删除的模块和测试不再可用，需保证新测试覆盖完整。
- 影响：对使用 harmony 模型（如 gpt-oss）的用户，流式行为将改善（修复了之前的 bug），非流式行为不变。对系统无性能影响。开发团队可获得更清晰的代码结构，便于后续维护。
- 风险标记：核心路径变更，删除旧模块，流式路径重写，无实时 review

关联脉络

- PR #45171 [Refactor] Chat Completions Harmony Refactor, non-streaming path.: 前序非流式重构 PR，本 PR 是其流式后续，共同完成了统一 Parser 的完整重构。

- PR #37070 [Bugfix] Fix harmony streaming tool call crash and argument splitting: 关联的 bug issue，描述流式工具调用问题，本 PR 直接修复了这些问题。