# PR #44330 完整报告

vllm-project/vllm

[Bugfix] GPT-OSS instruction rendering

合并时间：2026-06-06 01:52

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44330

## 执行摘要

- 一句话：修复 GPT-OSS 指令渲染与 HF 模板不一致
- 推荐动作：
  - 值得精读：特别是 build_harmony_preamble 的设计和如何通过 extract_instructions_from_messages 抽象消息预处理逻辑。
  - 关注测试：四个 case 的渲染对比示例是验证行为的好材料。
  - 留意环境变量：VLLM_GPT_OSS_HARMONY_SYSTEM_INSTRUCTIONS 的行为需在部署时确认。

## 功能与动机

根据 issue #33210，GPT-OSS 的指令渲染与 HF 的 apply_chat_template 不一致：vLLM 总是插入一个默认的 developer 消息，且用户的 system/developer 消息作为单独的消息放在后面。本 PR 修复此问题，将第一个 system/developer 指令消息折叠到 preamble 的 developer 块中，并确保在无指令无工具时跳过 developer 块。

## 实现拆解

1. 新增 extract_instructions_from_messages 函数（位于 harmony_utils.py）：从消息列表中分离出首条 system/developer 消息的文本内容，并返回剩余消息列表。
2. 新增 build_harmony_preamble 函数（位于 harmony_utils.py）：构建包含 system 消息和 developer 消息的 preamble，将指令、工具描述、浏览器 / 代码解释器等内置工具描述合并到 developer 块中。
3. 改造 Chat Completions 入口（render/serving.py 的 _make_request_with_harmony）：原逻辑固定创建 system 消息 + 可选的 developer 消息（仅当有工具时）。现改为调用 extract_instructions_from_messages 并传入 build_harmony_preamble，使首条 system/developer 指令自然融入 preamble。
4. 改造 Responses API 入口（responses/serving.py 的 _construct_input_messages_with_harmony）：类似地，先尝试从 request.instructions 获取指令，否则从 request.input 中提取首条指令消息，再调用 build_harmony_preamble。原冗长的 _construct_harmony_system_input_message 被拆分为更轻量的 _get_harmony_builtin_tool_descriptions。

5. 统一消息转换逻辑（responses/harmony.py 的 _parse_chat_format_message 和 response_input_to_harmony）：将 system/developer 消息统一转换为带有 DeveloperContent 的消息，通过 get_system_or_developer_message 根据环境变量 VLLM_GPT_OSS_HARMONY_SYSTEM_INSTRUCTIONS 决定使用 system 还是 developer 角色。
6. 配套测试：新增 test_system_message_without_tools 和 test_system_message_with_tools 验证 Chat Completions 路径；更新 test_response_input_to_harmony.py 中 developer 消息的预期输出；添加 test_developer_message 到渲染测试中。

关键文件：

- vllm/entrypoints/openai/parser/harmony_utils.py（模块 入口解析；类别 source；类型 dependency-wiring；符号 get_system_or_developer_message, flatten_input_text_content, extract_instructions_from_messages, build_harmony_preamble）：核心 utils，新增 extract_instructions_from_messages、build_harmony_preamble 等关键函数，重构消息预处理逻辑
- vllm/entrypoints/openai/responses/serving.py（模块 入口层；类别 source；类型 core-logic；符号 _get_harmony_builtin_tool_descriptions, _construct_input_messages_with_harmony）：Responses API 入口重构，提取指令和工具描述逻辑，使用新的 build_harmony_preamble
- vllm/entrypoints/openai/responses/harmony.py（模块 入口层；类别 source；类型 core-logic；符号 _parse_chat_format_message, response_input_to_harmony）：统一 system/developer 消息转换为 DeveloperContent，使用新工具函数
- vllm/entrypoints/serve/render/serving.py（模块 入口层；类别 source；类型 core-logic；符号 _make_request_with_harmony）：Chat Completions 入口使用新的 extract_instructions + build_harmony_preamble 替换原有直接创建 system/developer 消息
- tests/entrypoints/openai/chat_completion/test_serving_chat.py（模块 对话服务；类别 test；类型 test-coverage；符号 test_system_message_without_tools, test_system_message_with_tools）：新增 test_system_message_without_tools 和 test_system_message_with_tools 验证 system 消息正确折叠
- tests/entrypoints/openai/responses/test_response_input_to_harmony.py（模块 响应解析；类别 test；类型 test-coverage；符号 test_system_message, test_developer_message_gets_instructions_prefix, test_developer_message_array_content_concatenated）：更新 developer 消息的预期输出：使用 DeveloperContent 而非文本前缀
- tests/entrypoints/openai/parser/test_harmony_render_parity.py（模块 测试对齐；类别 test；类型 test-coverage；符号 test_developer_message）：新增 test_developer_message 验证渲染与 HF 一致
- tests/entrypoints/openai/parser/test_harmony_utils.py（模块 测试工具；类别 test；类型 test-coverage）：适配 extract_instructions_from_messages 和 flatten_input_text_content 的改动

- tests/entrypoints/openai/utils.py（模块 测试公用；类别 test；类型 test-coverage）：辅助测试文件，增加对 DeveloperContent 的校验支持

关键符号：get_system_or_developer_message, flatten_input_text_content, extract_instructions_from_messages, build_harmony_preamble, _construct_input_messages_with_harmony, _make_request_with_harmony, _parse_chat_format_message, response_input_to_harmony

## 关键源码片段

### vllm/entrypoints/openai/parser/harmony_utils.py

核心 utils，新增 extract_instructions_from_messages、build_harmony_preamble 等关键函数，重构消息预处理逻辑

```
# harmony_utils.py — 新增的核心函数

def extract_instructions_from_messages(
    messages: Sequence[Any],
) -> tuple[str | None, list[Any]]:
    """
    从消息列表中分离出首条 system/developer 消息的文本内容。
    返回（指令文本，剩余消息列表）。如果首条不是 system/developer，返回 (None, messages)。
    """
    remaining_messages = list(messages)
    if not remaining_messages:
        return None, remaining_messages

    first_message = remaining_messages[0]
    if not isinstance(first_message, dict):
        if hasattr(first_message, "to_dict"):
            first_message = first_message.to_dict()
        elif hasattr(first_message, "model_dump"):
            first_message = first_message.model_dump(exclude_none=True)
        else:
            raise ValueError(f"Unknown message type: {type(first_message)}")

    if first_message.get("role") not in ("system", "developer"):
        return None, remaining_messages

    instructions = flatten_input_text_content(first_message.get("content"))
    return instructions, remaining_messages[1:]


def build_harmony_preamble(
    *,
    instructions: str | None = None,
    tools: list[Tool | ChatCompletionToolsParam] | None = None,
    reasoning_effort: str | None = None,
    browser_description: str | None = None,
```

```
    python_description: str | None = None,
    container_description: str | None = None,
    with_custom_tools: bool = False,
) -> list[Message]:
    """
    构建 Harmony 请求的 preamble：system 消息 + 可选的 developer
    消息（包含指令和工具描述）。
    如果既没有 instructions 也没有 tools，且无内置工具，则只返回 system 消息。
    """
    sys_msg = get_system_message(
        reasoning_effort=reasoning_effort,
        browser_description=browser_description,
        python_description=python_description,
        container_description=container_description,
        with_custom_tools=with_custom_tools,
    )
    # 只有当有指令或工具时才添加 developer 消息
    if instructions or tools or browser_description or python_description or container_description:
        dev_msg = get_developer_message(
            instructions=instructions,
            tools=tools,
        )
        return [sys_msg, dev_msg]
    return [sys_msg]
```

## vllm/entrypoints/openai/responses/serving.py

Responses API 入口重构，提取指令和工具描述逻辑，使用新的 build_harmony_preamble

```
# responses/serving.py — _construct_input_messages_with_harmony 关键重构部分

def _construct_input_messages_with_harmony(
    self,
    request: ResponsesRequest,
    prev_response: ResponsesResponse | None,
) -> list[OpenAIHarmonyMessage]:
    messages: list[OpenAIHarmonyMessage] = []
    request_input = request.input

    if prev_response is None:
        # 新的对话：提取指令，构建 preamble
        tool_types = extract_tool_types(request.tools)
        with_custom_tools = has_custom_tools(tool_types)

        instructions = request.instructions
        if instructions is None and isinstance(request_input, list):
            # 从 input 中提取首条 system/developer 消息作为指令
            instructions, request_input = extract_instructions_from_messages(request_input)

        tool_descriptions = self._get_harmony_builtin_tool_descriptions(request, tool_types)
```

```
        # 构建 Harmony preamble（system + developer 消息）
        preamble = build_harmony_preamble(
            instructions=instructions,
            tools=request.tools if with_custom_tools else None,
            browser_description=tool_descriptions["browser_description"],
            python_description=tool_descriptions["python_description"],
            container_description=tool_descriptions["container_description"],
            with_custom_tools=with_custom_tools,
        )
        messages.extend(preamble)

        # 转换剩余的输入消息（已去除首条指令消息）
        messages.extend(response_input_to_harmony_list(request_input))
    else:
        # 已有对话：直接转换所有消息
        messages = response_input_to_harmony_list(request_input)
        messages = construct_harmony_previous_input_messages(prev_response, messages)
    return messages
```

## 评论区精华

reviewer bbrowning 在简化 review 中表示“同意与官方 chat template 对齐，因为模型供应商的 apply_chat_template 是事实标准”，并感谢 PR 描述中的 side-by-side 对比帮助理解变更影响。没有其他争议讨论。

- Review 批准和对齐 HF 的讨论 (other): 无争议，直接合并。

## 风险与影响

- 风险：
  - 渲染行为变化：现有依赖原始渲染格式的客户端可能受到影响，但新行为更接近 HF 预期，属于修复性变更。
  - 环境变量依赖：VLLM_GPT_OSS_HARMONY_SYSTEM_INSTRUCTIONS 控制是否使用 system 角色，如果用户依赖该变量但未更新部署，可能导致意外行为。测试已覆盖两种模式。
  - 工具折叠逻辑：工具描述与指令合并到同一 developer 块，若工具列表为空则无 developer 块，可能改变已有工具调用的输出格式。但该逻辑与 HF 保持一致。
  - Responses API 路径：_construct_input_messages_with_harmony 重构后，request.instructions 优先级高于从 input 提取，可能影响使用指令的客户端。但这是明确的设计选择，文档已更新。
- 影响：
  - 用户影响：所有使用 GPT-OSS 模型的 Chat Completions 和 Responses API 用户。生成的消息序列改变，但对齐 HF 后模型输出质量有望提升。
  - 系统影响：无性能影响；改动集中在入口解析层，不涉及核心引擎。
  - 团队影响：维护者需注意未来新增模型时遵循此模式。

- 风险标记：渲染逻辑变更，环境变量依赖，工具折叠逻辑，Responses API 重构

## 关联脉络

- 暂无明显关联 PR