PR #47550 完整报告

vllm-project/vllm

[ROCm][CI][Bugfix] Fix flaky parallel tool-call streaming (test assertion + Mistral/Granite parsers)

合并时间：2026-07-07 07:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47550

执行摘要

- 一句话：修复并行 tool-call 流式测试断言和 Mistral/Granite Parser bug
- 推荐动作：此 PR 值得精读，尤其关注流式场景下隐式假设的识别和修复方式，以及 parser 状态机设计。对理解 OpenAI 流式 delta 格式和测试 flakiness 根因分析很有帮助。

功能与动机

AMD CI 中 test_parallel_tool_calls 间歇性失败，原因为测试断言 assert len(streamed_tool_calls) == 1 在 stream_interval>1 时不再成立。进一步分析发现 Mistral 和 Granite parser 在遇到多 token chunk 时分别将并行调用合并到同一个索引或丢失第二个调用的名称，属于实际的数据损坏。详见 PR body。

实现拆解

1. 修复测试断言：在 tests/tool_use/test_parallel_tool_calls.py 中，将单条 delta 的断言改为遍历所有 delta，并增加 index 非递减检查。
2. 修复 Mistral Parser：在 vllm/tool_parsers/mistral_tool_parser.py 中，增加 starting_new_tool 标记，在 ijson start_map 事件时置位，替代原来依靠前后 streaming state 比较来检测新 tool 的方法（该方法在 batched delta 下失效）。移除了旧的状态比较逻辑，直接使用该标记。
3. 修复 Granite Parser：在 vllm/tool_parsers/granite_tool_parser.py 中，增加 current_tool_name_sent 判断，确保只有当前工具的名称已发出后才推进到下一个工具（因为 Granite 先发 arguments 后发 name）。同时将 current_tool_id 改为 +=1 递增，避免跳跃。
4. 修复 thinking_token_budget 测试：在 tests/entrypoints/openai/chat_completion/test_thinking_token_budget.py 中，从计数 reasoning chunk 改为按 token id 统计 reasoning 范围内的 token 数，通过 return_token_ids 获取 token id 流并使用 _count_reasoning_decode_token_ids_between_markers 计算。
5. 添加回归测试：在 tests/tool_parsers/test_granite_tool_parser.py 和 tests/tool_parsers/test_mistral_tool_parser.py 中，新增使用 split_string_into_token_deltas 生成模拟 chunk 并指定 chunk_size 参数（mistral 中通过 stream_delta_message_generator 的 chunk_size）的测试用例，确保多个 chunk 边界下 parser 行为正确。

关键文件：

- tests/tool_parsers/test_granite_tool_parser.py（模块 工具解析器；类别 test；类型 test-coverage；符号 granite_tokenizer, test_streaming_parallel_calls_batched_deltas）：添加针对 Granite parser 的 batched delta 回归测试，覆盖并行调用边界。
- tests/tool_parsers/test_mistral_tool_parser.py（模块 工具解析器；类别 test；类型 test-coverage；符号 test_streaming_pre_v11_parallel_calls_batched_deltas）：修改 stream_delta_message_generator 支持 chunk_size，并添加并行调用 batched delta 回归测试。
- vllm/tool_parsers/mistral_tool_parser.py（模块 工具解析器；类别 source；类型 core-logic）：核心修复：在 update_stream_state_pre_v11_tokenizer 中添加 starting_new_tool 标记，替代状态比较来检测新 tool 开始。
- vllm/tool_parsers/granite_tool_parser.py（模块 工具解析器；类别 source；类型 core-logic）：核心修复：增加 current_tool_name_sent 条件防止在名称未发出时推进到下一工具。
- tests/entrypoints/openai/chat_completion/test_thinking_token_budget.py（模块 预算测试；类别 test；类型 test-coverage）：修复 thinking_token_budget 测试中的计数方式，从 chunk 计数改为 token id 计数。
- tests/tool_use/test_parallel_tool_calls.py（模块 并行测试；类别 test；类型 test-coverage）：修复 flaky 测试断言，允许一个 streamed chunk 包含多个 tool-call delta。

关键符号：GraniteToolParser.extract_tool_calls_streaming, MistralToolParser.update_stream_state_pre_v11_tokenizer, MistralToolParser._extract_tool_calls_streaming_pre_v11_tokenizer, stream_delta_message_generator, test_streaming_parallel_calls_batched_deltas, test_streaming_pre_v11_parallel_calls_batched_deltas, test_thinking_token_budget_limits_reasoning

关键源码片段

tests/tool_parsers/test_granite_tool_parser.py

添加针对 Granite parser 的 batched delta 回归测试，覆盖并行调用边界。

```
import json
import pytest

from tests.tool_parsers.common_tests import ToolParserTestConfig, ToolParserTests
from tests.tool_parsers.utils import run_tool_extraction, run_tool_extraction_streaming, split_
string_into_token_deltas
from vllm.tokenizers import get_tokenizer
from vllm.tool_parsers.granite_tool_parser import GraniteToolParser


# Granite emits arguments before name and its own tokenizer (not gpt2) is used
# here so the token boundaries match production; get_tokenizer only fetches the
```

```
# small tokenizer files, not the model weights.
@pytest.fixture(scope="module")
def granite_tokenizer():
    return get_tokenizer(tokenizer_name="ibm-granite/granite-3.1-8b-instruct")


@pytest.mark.parametrize("chunk_size", [2, 3, 4, 5])
def test_streaming_parallel_calls_batched_deltas(granite_tokenizer, chunk_size):
    """A batched delta (multiple tokens) spanning the boundary between two
    parallel calls must not drop the first call's name. granite streams
    arguments before name, so the name only completes as the next call appears.
    """
    parser = GraniteToolParser(granite_tokenizer)
    model_output = (
        '<|tool_call|> [{"arguments": {"city": "Tokyo"}, "name": "get_weather"}, '
        '{"arguments": {"timezone": "Asia/Tokyo"}, "name": "get_time"}]'
    )
    token_deltas = split_string_into_token_deltas(granite_tokenizer, model_output)
    batched = [
        "".join(token_deltas[i : i + chunk_size])
        for i in range(0, len(token_deltas), chunk_size)
    ]
    reconstructor = run_tool_extraction_streaming(
        parser, batched, assert_one_tool_per_delta=False
    )
    names = [tc.function.name for tc in reconstructor.tool_calls]
    assert names == ["get_weather", "get_time"]
    # Trailing args of the final call are flushed by the serving layer
    assert json.loads(reconstructor.tool_calls[0].function.arguments) == {
        "city": "Tokyo"
    }
```

tests/tool_parsers/test_mistral_tool_parser.py

修改 stream_delta_message_generator 支持 chunk_size，并添加并行调用 batched delta 回归测试。

```
def stream_delta_message_generator(
    mistral_tool_parser: MistralToolParser,
    mistral_tokenizer: TokenizerLike,
    model_output: str | None,
    tools: list[tuple[str, str]] | None,
    chunk_size: int = 1, # 新增参数，控制每个 streamed delta 包含的 token 数量
) -> Generator[DeltaMessage, None, None]:
    if (
        isinstance(mistral_tokenizer, MistralTokenizer)
        and mistral_tokenizer.version >= 11
    ):
        assert tools is not None
        assistant_msg = AssistantMessage(
```

```
            tool_calls=[
                ToolCall(
                    function=FunctionCall(
                        name=name,
                        arguments=arg,
                    )
                )
                for (name, arg) in tools
            ],
        )
        request = InstructRequest(
            messages=[assistant_msg],
        )
        all_token_ids = mistral_tokenizer.instruct.encode_instruct(request).tokens
    else:
        assert model_output is not None
        all_token_ids = mistral_tokenizer.encode(model_output, add_special_tokens=False)

    all_token_ids = fix_tool_call_tokenization(
        all_token_ids, mistral_tool_parser, mistral_tokenizer
    )

    previous_text = ""
    previous_tokens = None
    prefix_offset = 0
    read_offset = 0
    pending_text = "" # 新增：累积文本
    pending_token_ids: list[int] = [] # 新增：累积 token id
    for i, delta_token in enumerate(all_token_ids):
        (new_tokens, delta_text, new_prefix_offset, new_read_offset) = (
            detokenize_incrementally(
                tokenizer=mistral_tokenizer,
                all_input_ids=all_token_ids[: i + 1],
                prev_tokens=previous_tokens,
                prefix_offset=prefix_offset,
                read_offset=read_offset,
                skip_special_tokens=isinstance(mistral_tokenizer, MistralTokenizer),
                spaces_between_special_tokens=True,
            )
        )
        previous_tokens = (
            previous_tokens + new_tokens if previous_tokens else new_tokens
        )
        prefix_offset = new_prefix_offset
        read_offset = new_read_offset

        # Buffer tokens so each streamed delta can carry ``chunk_size`` tokens,
        # reproducing the multi-token deltas produced by async scheduling /
        # stream_interval > 1.
```

```
        pending_text += delta_text
        pending_token_ids.append(delta_token)
        if len(pending_token_ids) < chunk_size and i != len(all_token_ids) - 1:
            continue

        previous_token_ids = all_token_ids[: i + 1 - len(pending_token_ids)]
        current_token_ids = all_token_ids[: i + 1]
        current_text = previous_text + pending_text

        delta_message = mistral_tool_parser.extract_tool_calls_streaming(
            previous_text,
            current_text,
            pending_text,
            previous_token_ids,
            current_token_ids,
            pending_token_ids,
            request=_DUMMY_REQUEST,
        )
        if delta_message:
            yield delta_message

        previous_text = current_text
        pending_text = ""
        pending_token_ids = []
```

评论区精华

bbrowning 在 review 中评论："The assertions here used to be accurate before async scheduler, spec decoding, stream_interval, and similar that can lead to more than 1 token per delta. But, they were always testing a false invariant (one tool call per delta) that just happened to be true before. Thanks for fixing this and keeping the CI flakes under control!" 表明原来测试假设现在已不成立，修正是必要的。

- 测试断言不变式不再成立 (testing): 修改断言，遍历所有 delta 并检查 index 非递减。

风险与影响

- 风险：parser 的修改影响流式 tool-call 提取核心逻辑，可能对非并行调用或单 token delta 场景有副作用，但通过添加回归测试覆盖了多 chunk 边界，风险降低。thinking_token_budget 测试依赖 return_token_ids 参数，需确认该参数在所用模型上可用（默认关闭），否则测试可能跳过或失败。整体风险较低。
- 影响：修复 AMD CI 上多个 flaky 测试（test_parallel_tool_calls、test_thinking_token_budget_limits_reasoning、test_parallel_tool_calls_false 等），确保并行 tool-call 流式场景下 Mistral 和 Granite 模型的数据正确性。用户无感知。
- 风险标记：流式 tool-call 核心路径变更，并行调用索引完整性，语法解析状态机调整

关联脉络

- PR #47606 后续修复 remaining flake test_named_tool_use: PR 作者在评论区提到还有一个 remaining flake 在 follow-up PR #47606 中修复。