

PR #41131 完整报告

vllm-project/vllm

[Bugfix] Changed speech to text chunk timestamp to cumulative approach

合并时间: 2026-07-28 01:06

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/41131>

执行摘要

- 一句话: 语音转录长音频时间戳漂移修复
- 推荐动作: 核心 fix 思路清晰正确, review 过程中的精度讨论值得关注。建议 merge 前确保所有测试通过, 并关注是否有外部消费者依赖旧签名。

功能与动机

Issue #32588 报告了长音频 (>30s) 转录时每个分段时间戳逐渐漂移 (每段约 0.5s), 10 段后误差可达 5s。原因是 `split_audio` 并非在准确 30s 处切割, 而是在最后 1s 内搜索低能量分裂点, 导致每个 chunk 实际长度略短于 30s。旧实现 `start_time = float(idx * chunk_size_in_s)` 未考虑这个差异, 误差逐段累积。

实现拆解

1. 核心偏移计算改为累积方式: 在 `vllm/entrypoints/speech_to_text/base/serving.py` 的 `_preprocess_speech_to_text` 方法中, 新增 `chunk_start_offsets` 列表, 遍历 `chunks[:-1]`, 每次累加上一个 chunk 的实际采样点数 `chunk.shape[-1] / sample_rate` 得到真实的起始时间, 替代原先只返回 `(engine_inputs, duration)` 的签名, 现在返回三元组 `(engine_inputs, duration, chunk_start_offsets)`。
2. 消费端使用真实偏移: 在 `_create_speech_to_text` 方法中, 将原来的 `start_time = float(idx * chunk_size_in_s)` 替换为 `start_time = chunk_start_offsets[idx]`, 直接从累积列表取值, 消除漂移。
3. seek 字段类型修正: `_get_verbose_segments` 中 `segment_class(seek=start_time)` 原为 `float`, 因累计偏移可能为浮点且 `seek` 应为整数位置, 改为 `seek=int(start_time)`, 只对 `seek` 取整, 不影响 `start/end` 的时间计算。
4. 文档修正: `docs/contributing/model/transcription.md` 中删除“重叠 chunk”描述, 改为说明 `overlap_chunk_second` 仅用于搜索窗口而非实际重叠。
5. 测试配套: 新增 `tests/entrypoints/speech_to_text/transcription/test_chunk_timestamp_offset.py`, 构造 29.5s、29.7s、5s 三个 chunk, 验证返回的 `offsets` 为累积真实值而非固定间隔; 更新 `test_speech_to_text_cancellation.py` 和 `test_transcription_inter_chunk_spacing.py` 中 `mock` 的签名, 使其返回新增的 `chunk_start_offsets` 参数。

关键文件:

- vllm/entrypoints/speech_to_text/base/serving.py (模块 语音服务; 类别 source; 类型 core-logic; 符号 _preprocess_speech_to_text, _create_speech_to_text, _get_verbose_segments) : 核心逻辑修改: 新增累积偏移计算, 变更返回签名, 修正 seek 类型
- tests/entrypoints/speech_to_text/transcription/test_chunk_timestamp_offset.py (模块 测试; 类别 test; 类型 test-coverage; 符号 test_chunk_offsets_are_cumulative_not_nominal) : 新增回归测试, 验证累积偏移计算正确且不等于旧固定间隔
- tests/entrypoints/speech_to_text/test_speech_to_text_cancellation.py (模块 测试; 类别 test; 类型 test-coverage) : 修改 mock 返回值签名以适配新的三元组返回
- tests/entrypoints/speech_to_text/transcription/test_transcription_inter_chunk_spacing.py (模块 测试; 类别 test; 类型 test-coverage) : 修改 mock 返回值签名以适应新的返回类型
- docs/contributing/model/transcription.md (模块 文档; 类别 docs; 类型 documentation) : 修正关于重叠 chunk 的文档描述

关键符号: _preprocess_speech_to_text, _create_speech_to_text, _get_verbose_segments

关键源码片段

vllm/entrypoints/speech_to_text/base/serving.py

核心逻辑修改: 新增累积偏移计算, 变更返回签名, 修正 seek 类型

```
async def _preprocess_speech_to_text(
    self,
    request: SpeechToTextRequest,
    audio_data: bytes,
    request_id: str,
) -> tuple[list[EngineInput], float, list[float]]:
    # Validate request
    request.language = self.model_cls.validate_language(request.language)
    request.to_language = (
        self.model_cls.validate_language(request.to_language)
        if request.to_language
        else None
    )

    if len(audio_data) / 1024**2 > self.max_audio_filesize_mb:
        raise VLLMValidationError(
            "Maximum file size exceeded",
            parameter="audio_filesize_mb",
            value=len(audio_data) / 1024**2,
        )

    # Run cpu intensive preprocess step in a separate thread pool executor.
    chunks, duration = await self._decode_and_chunk_speech_async(audio_data)

    # 核心修复: 计算每个 chunk 在原始音频中的真实起始偏移
```

```

# 初始偏移为 0.0, 之后每个 chunk 的起始 = 前一个偏移 + 前一个 chunk 的实际时长 (采样数 / 采样率)
chunk_start_offsets: list[float] = [0.0]
for chunk in chunks[:-1]:
    chunk_start_offsets.append(
        chunk_start_offsets[-1] + chunk.shape[-1] / self.asr_config.sample_rate
    )

# ... 语言检测 (不变)
if request.language is None and getattr(
    self.model_cls, "supports_explicit_language_detection", False
):
    request.language = await self._detect_language(
        chunks[0], f"{request_id}-lang_detect"
    )

# 构建 prompts (不变)
parsed_prompts: list[DictPrompt] = []
for chunk in chunks:
    stt_params = request.build_stt_params(
        audio=chunk,
        stt_config=self.asr_config,
        model_config=self.model_config,
        task_type=self.task_type,
    )
    prompt = self.model_cls.get_generation_prompt(stt_params)
    parsed_prompt = parse_model_prompt(self.model_config, prompt)
    parsed_prompts.append(parsed_prompt)

engine_inputs = await self.renderer.render_cmpl_async(parsed_prompts)

# 现在返回三元组, 包含每个 chunk 的起始偏移
return engine_inputs, duration, chunk_start_offsets

```

tests/entrypoints/speech_to_text/transcription/test_chunk_timestamp_offset.py

新增回归测试, 验证累积偏移计算正确且不等于旧固定间隔

```

"""Regression test for chunk timestamp offset drift in _preprocess_speech_to_text."""

```

```

from __future__ import annotations
from unittest.mock import AsyncMock, MagicMock, patch
import numpy as np
import pytest
from vllm.config.speech_to_text import SpeechToTextConfig
from vllm.entrypoints.speech_to_text.base.serving import SpeechToTextBaseServing

SR = 16_000
_PATCH = "vllm.entrypoints.speech_to_text.base.serving"

```

```

@pytest.mark.asyncio
async def test_chunk_offsets_are_cumulative_not_nominal():
    """
    chunk_start_offsets 必须是基于实际 chunk 长度的累积值,
    而不是旧版的 `idx * max_audio_clip_s`。
    当 split_audio 在 30s 前找到安静区域导致 chunk 变短时,
    旧公式会产生漂移, 新公式保持精确。
    """

    # 模拟 split_audio 可能产出的三种实际 chunk 长度 (均 < 30s)
    chunk_lengths = [int(29.5 * SR), int(29.7 * SR), int(5.0 * SR)]
    chunks = [np.zeros(n, dtype=np.float32) for n in chunk_lengths]
    duration = sum(chunk_lengths) / SR

    # 期待的真实累积偏移 (秒)
    expected_offsets = [0.0, 29.5, 29.5 + 29.7] # = [0.0, 29.5, 59.2]
    # 旧方法可能产生的错误偏移 (固定 30s 间隔)
    wrong_offsets = [0.0, 30.0, 60.0]

    # 构造 mock server 实例
    serving = SpeechToTextBaseServing.__new__(SpeechToTextBaseServing)
    serving._decode_and_chunk_speech_async = AsyncMock(return_value=(chunks, duration))
    serving.asr_config = SpeechToTextConfig(
        sample_rate=float(SR),
        max_audio_clip_s=30,
        overlap_chunk_second=1,
        min_energy_split_window_size=1600,
    )
    serving.max_audio_filesize_mb = 100.0
    serving.model_cls = MagicMock()
    serving.model_cls.validate_language.side_effect = lambda lang: lang
    serving.model_cls.supports_explicit_language_detection = False
    serving.model_cls.get_generation_prompt.return_value = {}
    serving.model_config = MagicMock()
    serving.task_type = "transcribe"
    serving.renderer = MagicMock()
    serving.renderer.render_cmpl_async = AsyncMock(
        return_value=[MagicMock()] * len(chunks)
    )

    request = MagicMock()
    request.language = "en"
    request.to_language = None
    request.response_format = "json"
    request.build_stt_params.return_value = MagicMock()

    with patch(f"{PATCH}.parse_model_prompt", return_value=MagicMock()):
        _, _, offsets = await serving._preprocess_speech_to_text(
            request=request,
            audio_data=b"\x00",

```

```
request_id="test",
)
```

```
# 断言累积偏移与期待一致，且不等于旧错误值
assert offsets == pytest.approx(expected_offsets, abs=1e-6)
assert offsets != pytest.approx(wrong_offsets, abs=1e-6)
```

评论区精华

- 精度丢失风险：dr75 和 depthfirst-app[bot] 指出原提交中使用 `int(chunk_start_offsets[idx])` 会丢弃小数部分（如 `29.5→29`），导致 `verbose_json` 的 `start/end` 时间依然偏差。作者修正为在 `_get_verbose_segments` 内部只对 `seek` 字段做 `int`，`start_time` 保留 `float`。
- 浮点累积 vs 整数累积：dr75 建议用整数采样点求和后再除采样率以避免浮点累积误差，但认为当前 `chunk` 数量有限时浮点误差可忽略。作者表示可进一步优化但保留当前方案。
- 测试 mock 签名兼容：yewentao256 指出 `test_non_streaming_cancel_advances_all_chunk_generators` 等测试中的 mock 未更新三返回值签名导致失败，作者已修正。
- 文档不准确：dr75 指出文档中关于“重叠 chunk”的描述与实际 `split_audio` 行为不符，作者已修改为“搜索窗口”说明。
- `start_time` 精度丢失：`int()` 截断浮点 (correctness)：作者修正为在 `_get_verbose_segments` 中仅对 `seek` 字段取整数，`start_time` 保留 `float` 传入。
- 浮点累积与整数累积孰优 (design)：作者保留浮点方案并说明可后续优化；其他 reviewer 未反对。
- 测试 mock 签名不匹配 (testing)：作者更新该文件及 `inter_chunk_spacing` 测试的 mock 签名。
- 文档关于重叠 chunk 描述有误 (documentation)：作者更新文档，改为描述 `search window`。

风险与影响

- 风险：
 - 签名兼容性：`_preprocess_speech_to_text` 返回类型从二元组变为三元组，任何直接调用该方法的代码（如测试或子类）需同步更新。当前仓库内已全部适配，但外部 fork 可能受影响。
 - 浮点累积误差：连续浮点加法在极端多 `chunk`（如数百段）时可能产生亚秒级偏移，但实际场景中 `chunk` 数量通常少于 20，风险极低。
 - `seek` 取整合理性：`seek=int(start_time)` 丢弃小数部分，但 `seek` 字段在 Whisper 协议中本意是整数索引位置，取整符合规范。
- 影响：
 - 用户影响：使用 vLLM Whisper 转录长音频（>30s）的用户的 `segment` 时间戳将恢复准确，不再需要客户端侧按每段 0.5s 手动修正。
 - 系统影响：无性能变化，仅计算偏移方式改变；代码可读性略有提升（累积逻辑比固定间隔更直观）。

- 团队影响：新增回归测试覆盖了此场景；文档同步更新避免后续贡献者误解 `split_audio` 行为。
- 风险标记：签名变更需同步调用方，浮点累积微小误差，`seek` 取整语义正确

关联脉络

- 暂无明显关联 PR