PR #22848 完整报告

sgl-project/sglang

[Feature] WebSocket streaming audio input for ASR

合并时间：2026-05-27 22:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22848

执行摘要

- 一句话：为 ASR 添加 WebSocket 实时音频输入端点
- 推荐动作：值得精读。该 PR 展示了如何在现有架构上集成实时双向协议，对理解 OpenAI Realtime 协议实现、异步状态机设计、跨路径一致性处理（HTTP SSE vs WebSocket）有较高学习价值。建议关注 session.py 的状态隔离设计和 streaming_asr.py 的 emitted_text 改动。

功能与动机

引自 PR body：'Real-time use cases (live captioning, voice assistants, meeting transcription) need the opposite direction: the server accepts audio as it arrives and pushes partial transcripts back as the speaker talks.' 该 PR 实现 RFC #22474 的 M1 阶段，解决增量音频输入的缺失。此前 #22089 仅支持完整文件上传后的流式输出。

实现拆解

1. 新增 realtime/ 包（protocol.py, session.py, handler.py）：定义 OpenAI 兼容的 Pydantic 协议模型、RealtimeConnection 状态机、WebSocket 入口。
2. 注册路由：http_server.py 添加 /v1/realtime WebSocket 路由。
3. 入口处理：handler.py 实现预连接校验（模型是否支持流式、并发限制），通过 asyncio.Semaphore 控制最大连接数，然后创建 RealtimeConnection 并运行事件循环。
4. 会话管理：session.py 的 RealtimeConnection 维护三个内部状态（_SessionConfig、_AudioState、_ItemState），处理 session.update、input_audio_buffer.append/commit/clear 事件，调用 process_asr_chunk 驱动 ASR 推理，并发送服务器事件（delta, completed 等）。支持预提交 delta 的 sglang 扩展。
5. 流式核心重构：streaming_asr.py 原 StreamingASRState 增加 emitted_text 和 _record_emit，修复前缀回滚时模型继续性问题；新增 normalize_whitespace、needs_space、process_asr_chunk 等函数统一 HTTP SSE 和 WebSocket 路径。
6. 配置与适配：server_args.py 增加 --asr-max-buffer-seconds（单会话缓冲上限）和 --asr-max-concurrent-sessions（全局并发上限）。transcription_adapters/base.py 增加 model_sample_rate 属性供重采样使用。
7. 测试：test_qwen3_asr.py 扩增多语言测试和 WebSocket 流程测试，包含 WER 验证。单元测试补充 needs_space 边界情况。

关键文件：

- python/sglang/srt/entrypoints/openai/realtime/session.py（模块 会话管理；类别 source；类型 core-logic；符号 _resample_to_target_rate, _pcm_to_wav, _parse_client_event, _SessionConfig）：核心状态机，负责会话生命周期、音频缓冲、ASR 推理协调和事件发送，包含 740 行新代码。
- python/sglang/srt/entrypoints/openai/realtime/handler.py（模块 入口控制；类别 source；类型 entrypoint；符号 _safe_send, _safe_close, _reject_before_session, handle_realtime_transcription）：WebSocket 入口点，处理连接接受、并发限制、前置校验和异常清理，是整个 /v1/realtime 的流量入口。
- python/sglang/srt/entrypoints/openai/realtime/protocol.py（模块 协议定义；类别 source；类型 core-logic；符号 AudioPCM, AudioPCMU, AudioPCMA, AudioTranscription）：定义 WebSocket 协议的消息模型，将 OpenAI 标准 schema 与 sglang 扩展（如多采样率支持）桥接。
- python/sglang/srt/entrypoints/openai/streaming_asr.py（模块 流式处理；类别 source；类型 core-logic；符号 _record_emit, normalize_whitespace, _is_cjk, needs_space）：流式 ASR 核心逻辑，修复前缀回滚、统一空格处理，新增 HTTP 和 WebSocket 共享的 process_asr_chunk 函数。
- test/manual/models/test_qwen3_asr.py（模块 ASR 测试；类别 test；类型 test-coverage；符号 _normalize_for_wer, _wer, _pcm16_from_audio_bytes, _stream_websocket_async）：手动测试文件，新增 WebSocket 流程测试、多语言 WER 验证，确保 OOM 稳定。
- python/sglang/srt/entrypoints/openai/serving_transcription.py（模块 转录服务；类别 source；类型 refactor；符号 handle_websocket）：调整 HTTP SSE 路径，集成 shared needs_space 和 process_asr_chunk，删除冗余逻辑。

关键符号：_resample_to_target_rate, _pcm_to_wav, _parse_client_event, RealtimeConnection.init, RealtimeConnection.run, handle_realtime_transcription, _safe_send, _safe_close, _reject_before_session, process_asr_chunk, normalize_whitespace, needs_space, StreamingASRState.get_prefix_text, StreamingASRState.update, StreamingASRState.finalize, StreamingASRState._record_emit

关键源码片段

python/sglang/srt/entrypoints/openai/realtime/protocol.py

定义 WebSocket 协议的消息模型，将 OpenAI 标准 schema 与 sglang 扩展（如多采样率支持）桥接。

```
"""Wire schema for Realtime WS transcription sessions."""

from __future__ import annotations

from typing import Literal, Optional, Union

from openai.types.realtime import SessionUpdateEvent as _SessionUpdateEvent
```

```
from openai.types.realtime.audio_transcription import (
    AudioTranscription as _AudioTranscription,
)
from openai.types.realtime.realtime_audio_formats import AudioPCM as _AudioPCM
from openai.types.realtime.realtime_audio_formats import AudioPCMA as _AudioPCMA
from openai.types.realtime.realtime_audio_formats import AudioPCMU as _AudioPCMU
from openai.types.realtime.realtime_transcription_session_audio import (
    RealtimeTranscriptionSessionAudio as _AudioCfg,
)
from openai.types.realtime.realtime_transcription_session_audio_input import (
    RealtimeTranscriptionSessionAudioInput as _AudioInputCfg,
)
from openai.types.realtime.realtime_transcription_session_create_request import (
    RealtimeTranscriptionSessionCreateRequest as _SessionCfg,
)
from pydantic import Field
from typing_extensions import Annotated

# 当客户端未指定采样率时的默认值（OpenAI SDK 固定为 24000）
DEFAULT_INPUT_SAMPLE_RATE = 24000

# 支持的输入采样率（扩展了 OpenAI 的仅 24000，覆盖常见浏览器和 ASR 模型）
SUPPORTED_INPUT_SAMPLE_RATES = (16000, 24000, 48000)


class AudioPCM(_AudioPCM):
    # 覆盖 SDK 强制 Literal[24000]，允许客户端指定任意值
    type: Literal["audio/pcm"] = "audio/pcm"
    rate: Optional[int] = None


class AudioPCMU(_AudioPCMU):
    type: Literal["audio/pcmu"] = "audio/pcmu"


class AudioPCMA(_AudioPCMA):
    type: Literal["audio/pcma"] = "audio/pcma"


AudioInputFormat = Annotated[
    Union[AudioPCM, AudioPCMU, AudioPCMA],
    Field(discriminator="type"),
]


class AudioTranscription(_AudioTranscription):
    # sglang 支持任意 ASR 模型名，不限制为 whisper-1 等，客户端可提供本地模型名
    model: Optional[str] = None
```

```
class TranscriptionSessionAudioInput(_AudioInputCfg):
    format: Optional[AudioInputFormat] = None
    transcription: Optional[AudioTranscription] = None


class TranscriptionSessionAudio(_AudioCfg):
    input: Optional[TranscriptionSessionAudioInput] = None


class TranscriptionSessionConfig(_SessionCfg):
    audio: Optional[TranscriptionSessionAudio] = None


class SessionUpdateEvent(_SessionUpdateEvent):
    session: TranscriptionSessionConfig
```

评论区精华

讨论集中在协议兼容性和内部设计：

- 协议选择：AgainstEntropy 建议直接复用 OpenAI Realtime 协议，SammLSH 同意并最终实现，仅保留少数 sglang 扩展。
- 适配器访问：Avoid accessing private _adapter，改为在 handle_realtime_transcription 参数中传入 adapter 和 tokenizer_manager。
- 重采样库：librosa 将被废弃，AgainstEntropy 建议替换为 torchaudio，SammLSH 已更新。
- •扩展点：关于client_model不匹配拒绝、空音频快速路径、并发容量等细节均被确认或实现。
- 协议选择：OpenAI Realtime 标准化 (design): 最终协议与 OpenAI 标准兼容，仅保留少数 sglang 扩展（预提交 delta）。
- 适配器访问方式 (design): handler 签名改为接收 adapter 和 tokenizer_manager，不再引用 OpenAIServingTranscription。
- 重采样库切换 (librosa -> torchaudio) (performance): 已替换为 torchaudio，单元测试通过。
- client_model 不匹配处理 (correctness): 添加校验，不匹配时发送 not_supported 错误。
- 空音频快速路径 (performance): 在 append 事件中检测空字符串直接跳过。

风险与影响

- 风险：主要风险：
  - 并发资源消耗：每个 WebSocket 连接维护缓冲区和中转状态，大量连接可能导致内存增加。--asr-max-concurrent-sessions 可限制，但可能拒绝合法请求。
  - 协议兼容性：预提交 delta 偏离 OpenAI 标准，客户端若严格遵循 OpenAI 文档可能遗漏这些事件。但最终 commit 顺序符合标准，因此影响有限。
  - 重采样质量：从 librosa 切换 torchaudio 可能对极低或极高采样率有细微差异，但项目已广泛使用 torchaudio。

- 状态机正确性：streaming_asr.py 的 get_prefix_text 从 confirmed_text 改为 emitted_text 修复回滚问题，但可能影响某些边缘场景。已通过测试验证。
- 影响：用户可通过 ws://host/v1/realtime 使用标准 OpenAI 客户端实现实时 ASR，无需轮询或文件上传。团队需维护新端点，后续 M2 将引入 Producer-Consumer 模型进一步优化。系统负载受并发限制保护。
- 风险标记：新 WebSocket 端点，并行连接资源占用，协议兼容性差异，状态机复杂度

关联脉络

- PR #22073 [Feature] Qwen3-ASR model support: Qwen3-ASR 模型支持是本 PR 的基础，本 PR 在此基础上添加实时输入流式能力。
- PR #22089 [Feature] Add chunk-based streaming ASR for Qwen3-ASR: 前序 HTTP SSE 流式输出，本 PR 借鉴其 streaming_asr.py 设计并修复前缀回滚问题。
- PR #22474 [RFC]: Real-Time Streaming Audio Input for ASR Models: 本 PR 实现了该 RFC 的 M1 阶段，定义协议和架构方向。