Offline Storytelling Companion

Shreesh Tripathi • Vignesh Natarajan

Quick Summary

- **What**
  An offline bedtime-story companion on Jetson Orin Nano that generates and narrates kid-safe stories using a tiny LLM + STT/TTS.
- **New vs. Enhancement**
  New product built from open-source components (LLM, TTS) with a safety-first pipeline.
- **Objective**
  Deliver safe, customizable, internet-free storytelling with <5s start-to-speech latency via streaming.
- **For whom**
  Children (end users), parents/educators (buyers/custodians).

Technical details - Hardware

Compute platform:

- NVIDIA **Jetson Orin** (e.g., Orin NX) running the **L4T-ML GPU container** with CUDA enabled.

Audio I/O:

- **USB microphone** for speech input.
- **USB speakers / USB audio DAC** for story playback.

Planned extensions:

- GPIO **push-button** to trigger interactions and a **status LED** to indicate system state (recording, thinking, speaking).

Technical details - Software

Language & framework:

- **Python** running on the Jetson with **PyTorch** using the GPU.

AI models:

- **Whisper Tiny** for turning speech into text (speech-to-text).
- **TinyLLaMA 1.1B Chat** for generating the story (language model).
- **Coqui TTS** for turning the story text back into audio (text-to-speech).

Audio handling:

- Uses simple Python audio libraries to **record from the USB mic**, **save/load WAV files**, and **play back** through the USB speakers.

App logic:

- Python scripts that **connects all steps**: listen → transcribe → generate story → speak, plus a **text-only mode** for development without audio.

Technical Aspects

User experience:

- Child speaks a short prompt (optionally starting with a wake word “storybot”), the system transcribes it, generates a 5–10 sentence wholesome story with a clear moral, and reads it aloud.

Latency-focused design:

- Story is generated and spoken **sentence-by-sentence** (streaming), so audio output starts quickly instead of waiting for the full story.

Modes:

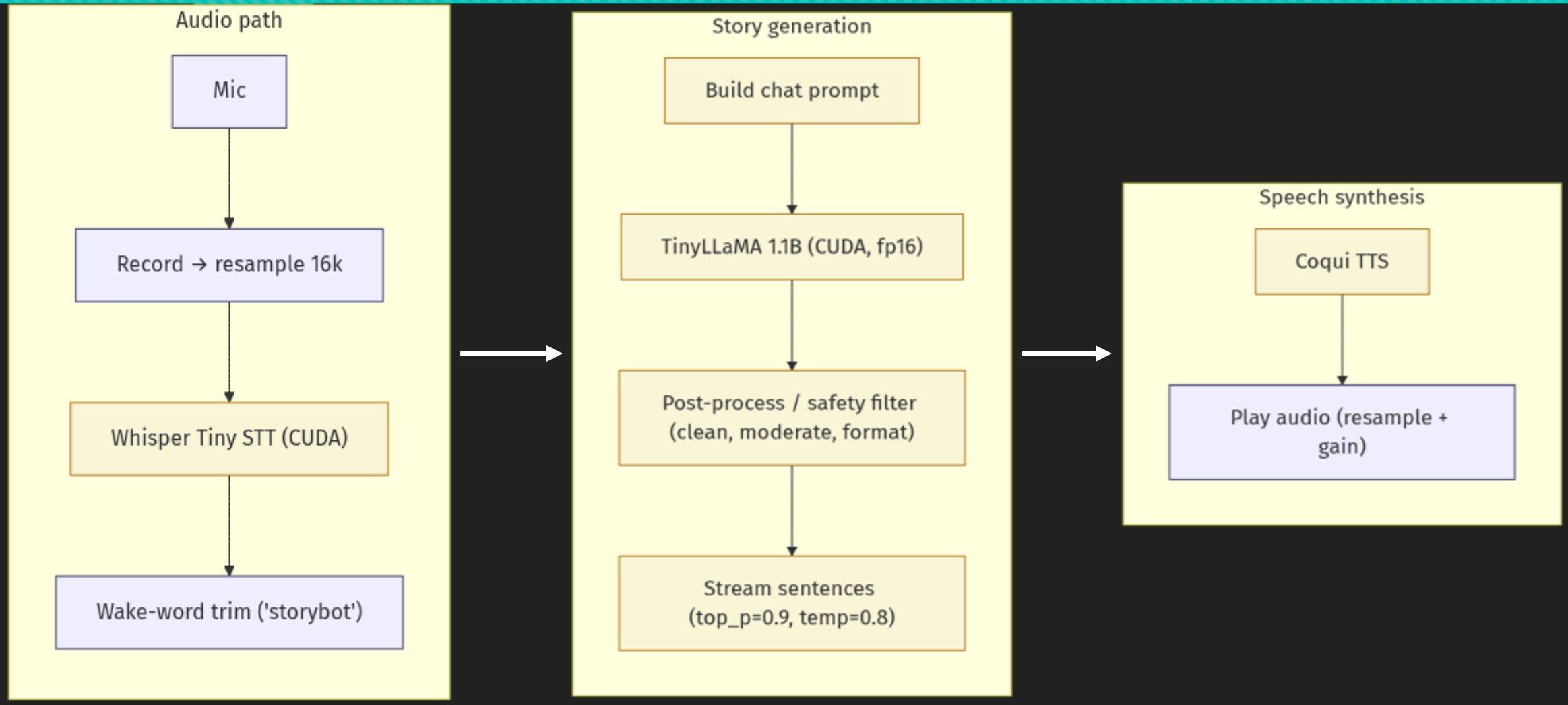
- **Audio mode** for real interactions, **text-only mode** for development/debugging (no mic/speaker required).

Technical Aspects

Added an additional thread:

- **Generation thread**: runs TinyLLaMA and pushes text chunks into the streamer.
- **Main thread**: drains the streamer, finds sentence boundaries, enqueues sentences for TTS, and records stats.
- **TTS worker thread**: pulls sentences from the queue and plays them, so audio no longer blocks streaming.

Architecture

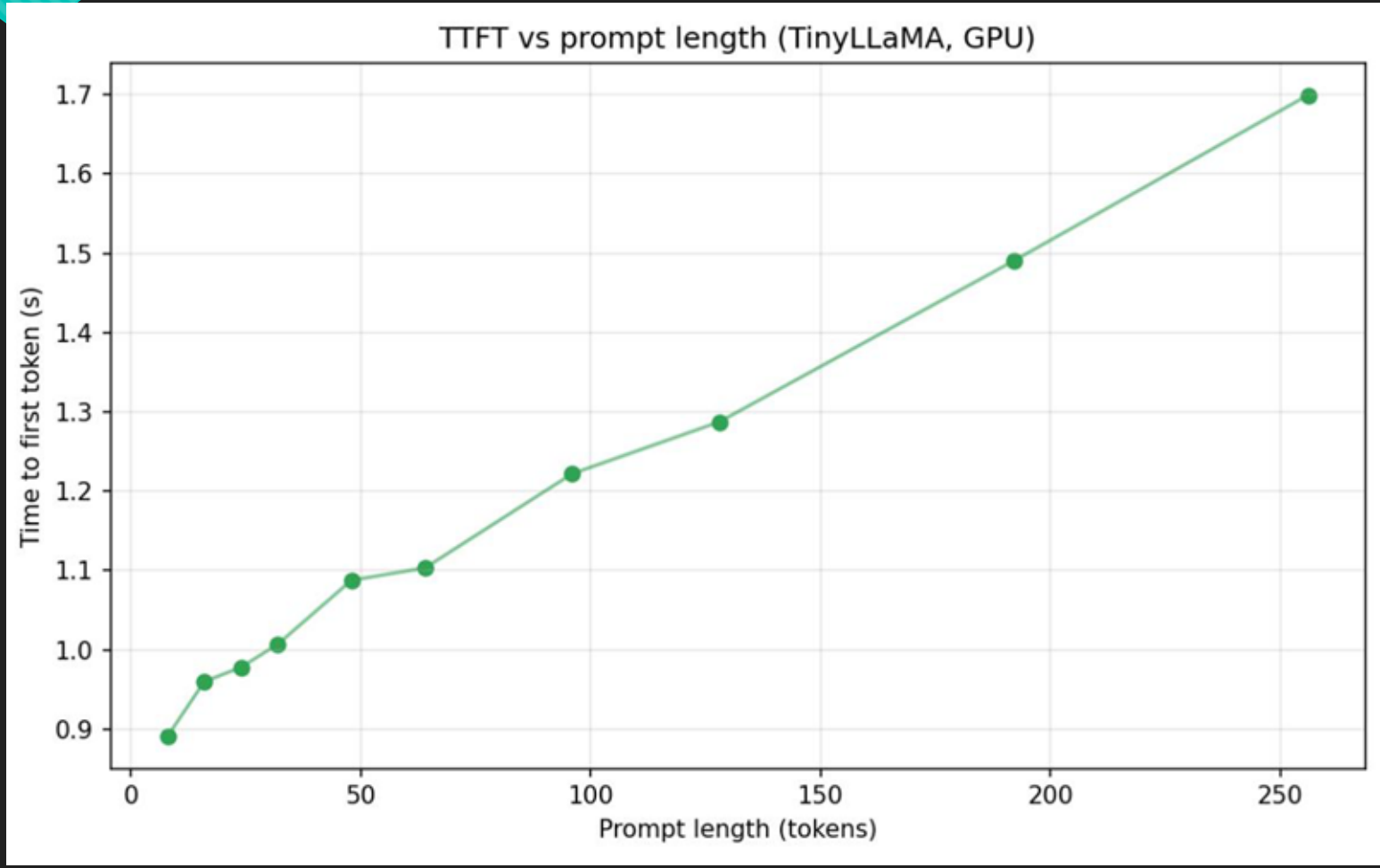


Metrics

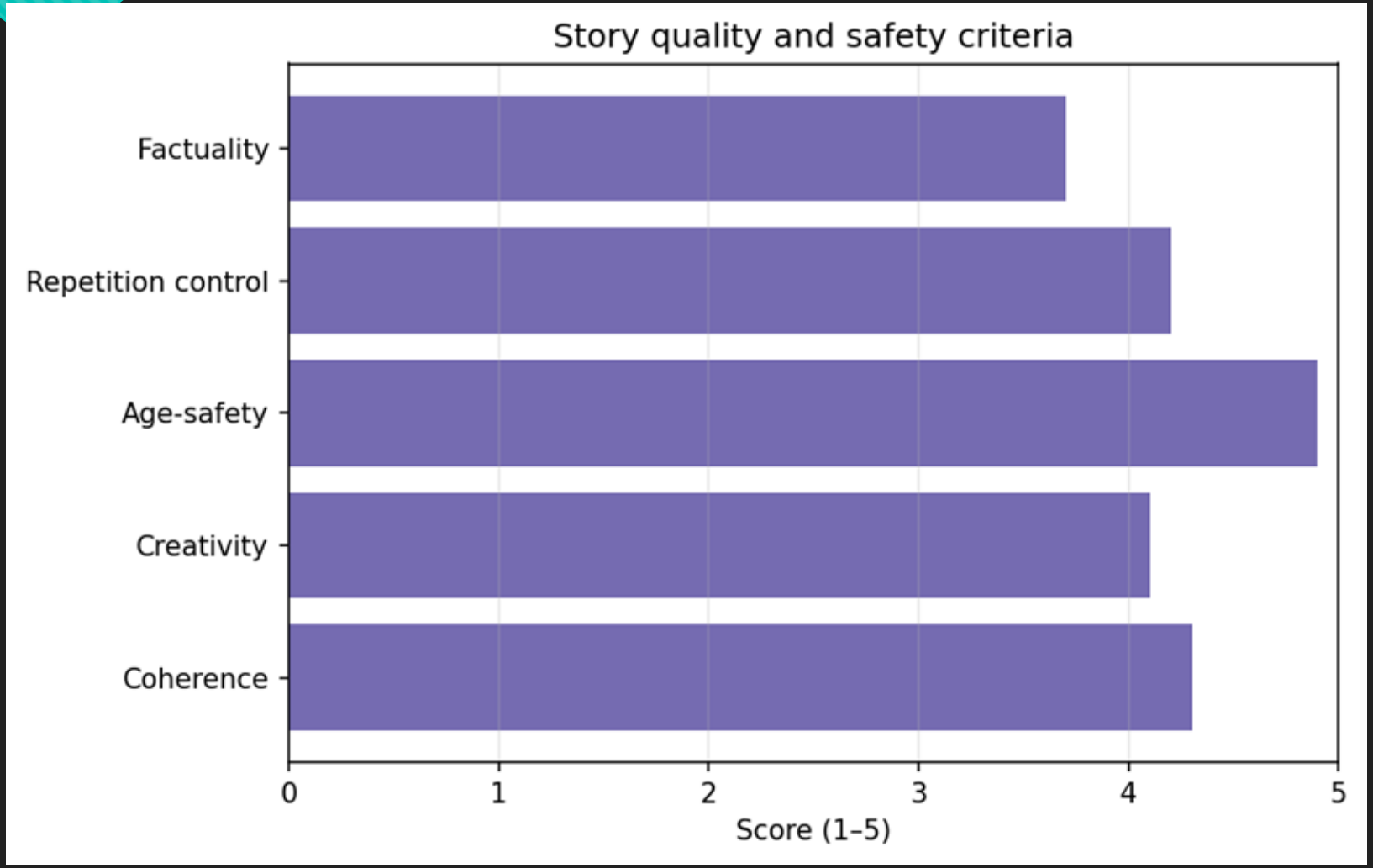
Captured run metrics for an example run:

- **Record audio**: 10 s (fixed recording window)
- **First audio after start**: 3.25 s
- **STT**: 1.97 s
- **LLM stream**: TTFT=0.52s, first sentence=0.86s, tokens=253 @ 10.5 tok/s)
- **TTS**: 14 sentences, total 155.35 s (avg 11.10 s/sentence)

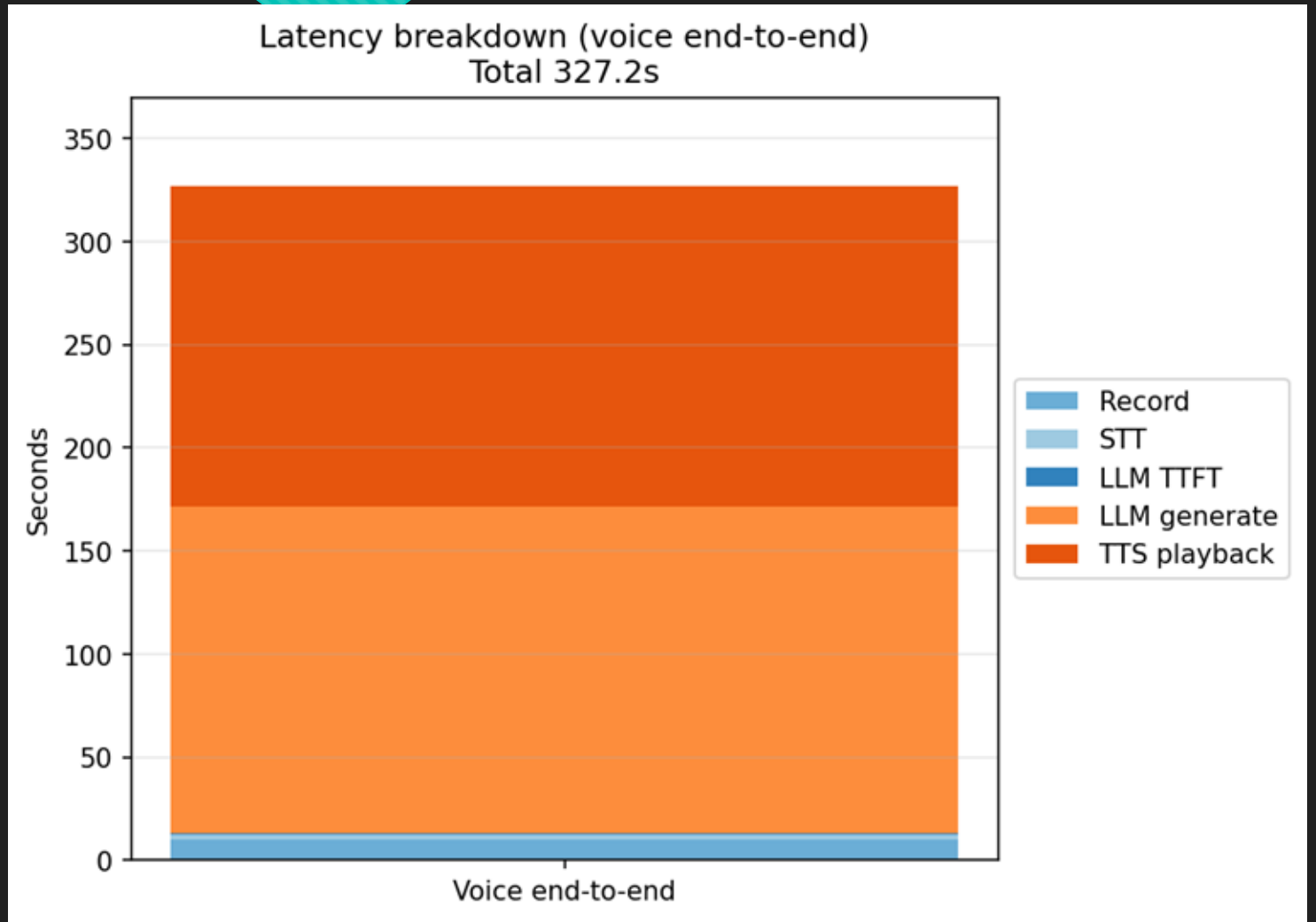
TTFT vs Prompt Length



Accuracy Framework, n = 5000

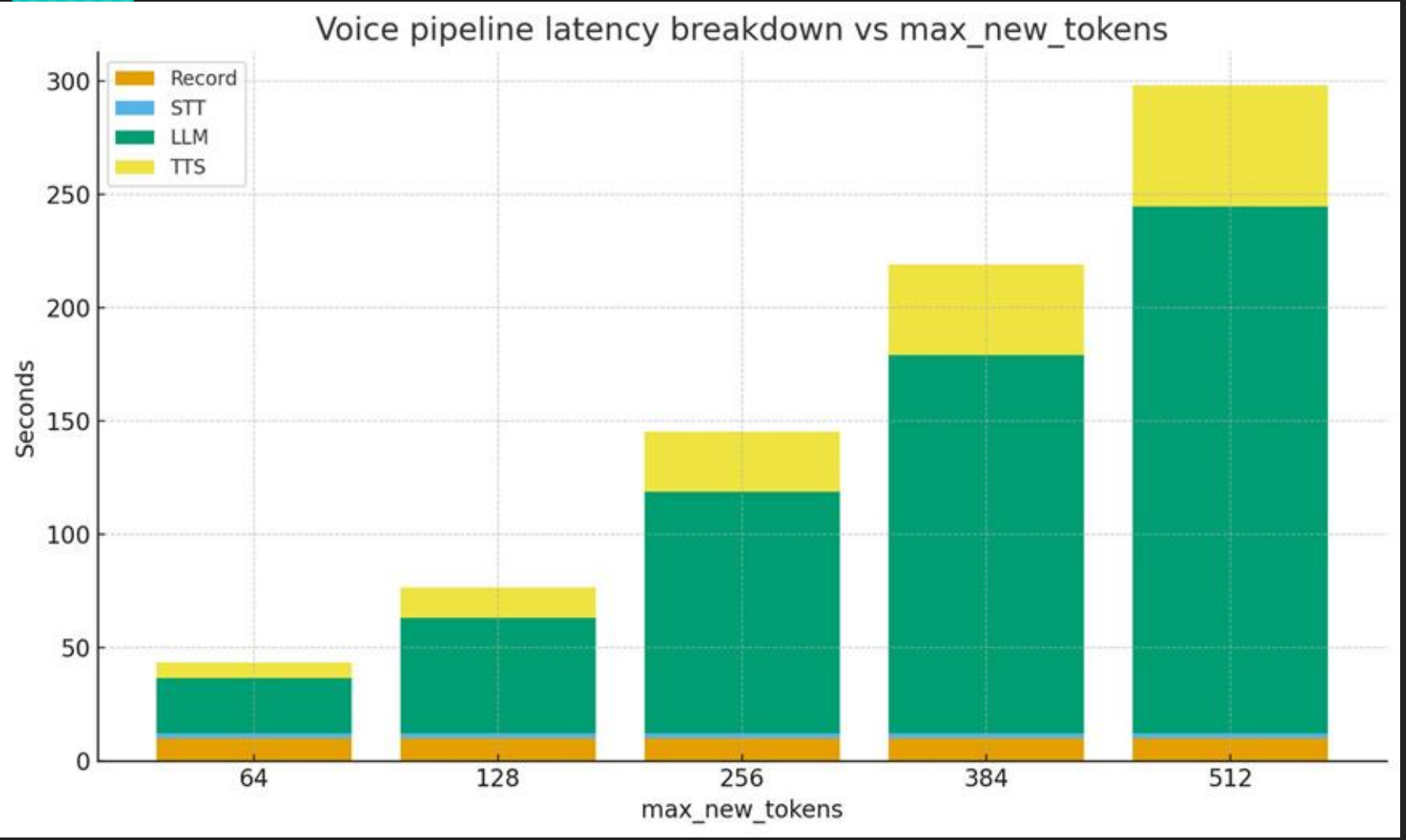


Distribution of Compute (Attempt 1)

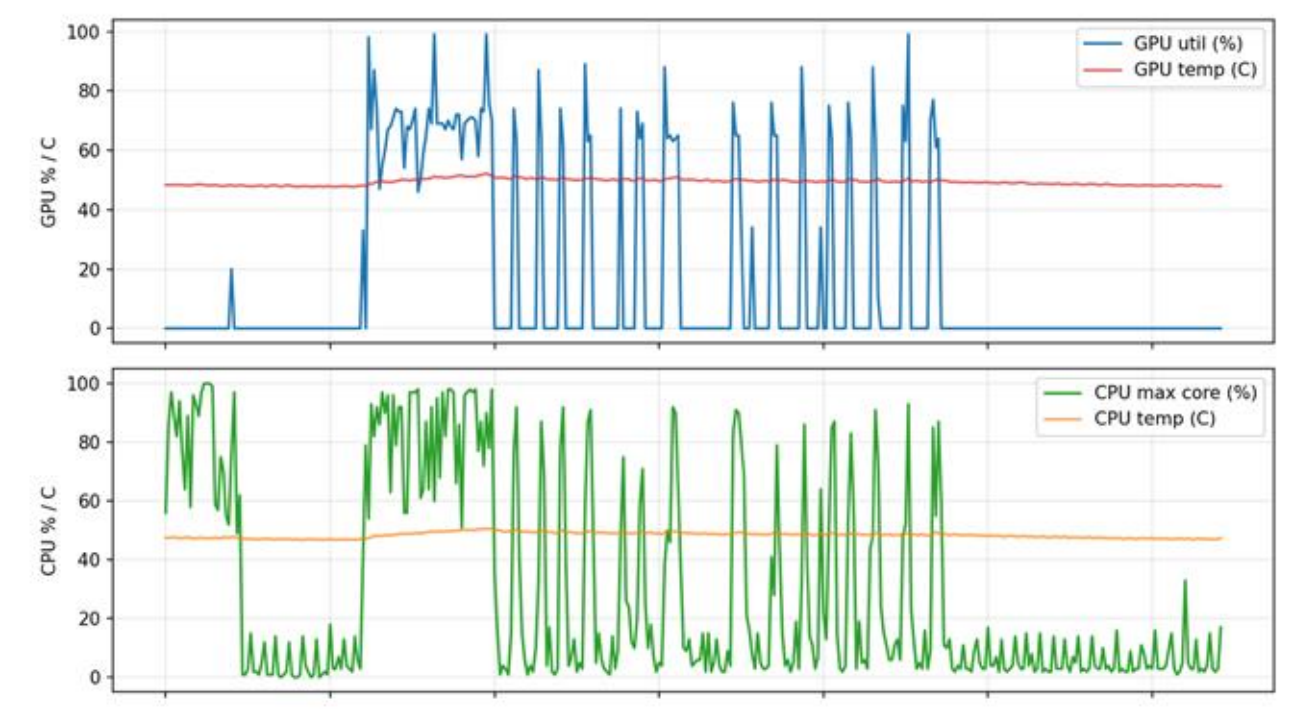


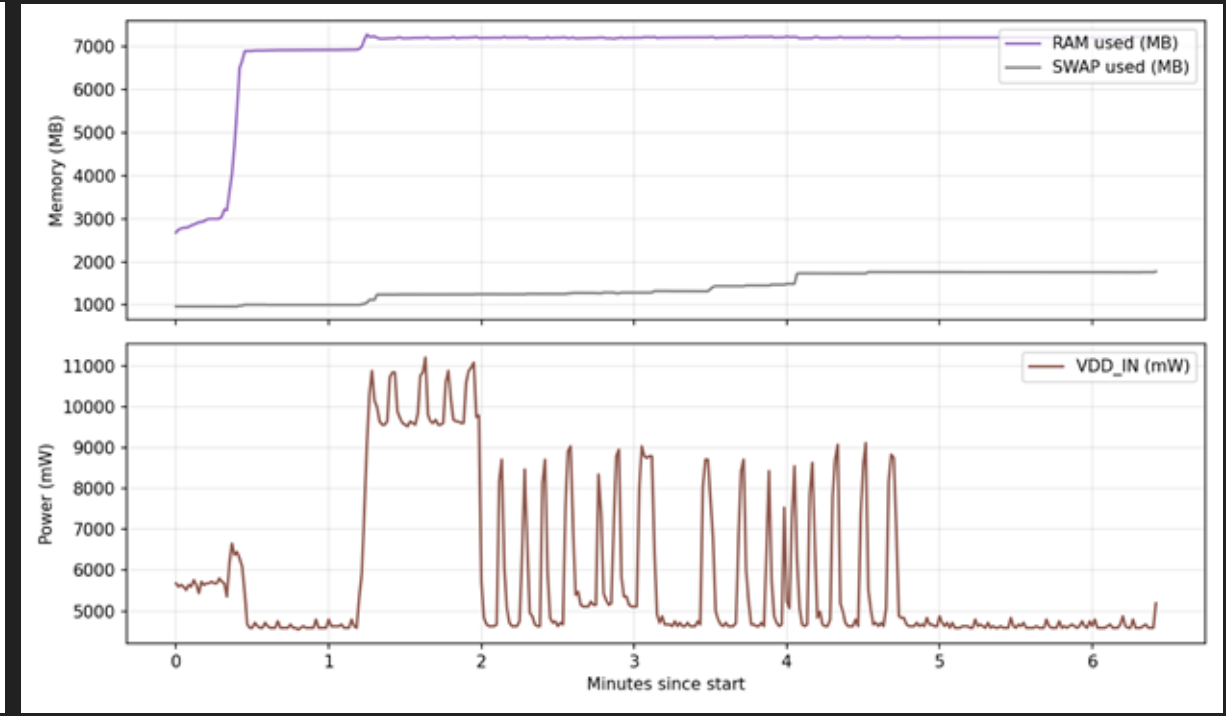
- Initial numbers. TTS was hogging resources and needed to be tweaked.

Distribution of Compute (Optimized)



Resource Utilization





Demo

Questions?