KubeEdge ASR Offloading
Signalogic is building an edge computing demo for KubeEdge

- Overview
- ASR Offloading
- Demo Capability
- Kaldi Interface
  - Data Flow
  - Software Architecture
- KubeEdge Integration
- Kaldi Info
  - Inference Performance
  - Integration
  - Architecture, DNNs
Overview

• **SigSRF packet + media processing software**
  – SigSRF is deployed by telecoms, LEAs, and analytics customers worldwide
  – extreme high capacity, robust packet interface, wideband audio codecs, jitter buffer and packet loss handling, stream alignment, etc.

• **ASR**
  – new, recently added to SigSRF
  – based on Kaldi

• **KubeEdge**
  – edge computing version of Kubernetes
  – open source owned, maintained by LF Edge

---

1 Linux Foundation
ASR Offloading

Edge Node Processing

Real-Time
- Wideband audio decoding
- Group calls
  - stream alignment
  - conferencing and merging
- ASR inference

Non Real-Time
- Diarization – not possible in real-time yet

Challenges
- Varying latencies between endpoints
- Background noise
- Overlapping conversations
- Avoid using GPUs when possible

Central DC Processing

Real-Time
- Larger models, multiple models

Non Real-Time
- Training
  - Augmentation for background noise / babble
- New vocabulary, speakers, language
Demo Capability

• **ASR based on Kaldi’s mini-librispeech model**
  – subset of librispeech model, which has 200k word vocabulary (English)
  – trained with fewer hours, producing a smaller model easier to use for development and testing
  – demo uses pre-trained x-vectors and i-vectors – no training required

• **SigSRF packet + media software**
  – voice/audio codecs - AMR, AMR-WB, EVS, G729, G726
  – RFCs - child streams (8108), DTMF (4733), 7198, others
  – concurrent sessions - 8 (demo subset of 512)
  – packet handling - jitter buffer, DTX, packet loss mitigation

• **Call groups (one or more endpoints)**
  – conferencing, merging, deduplication
  – ASR is applied to call group output
Kaldi Interface

• **Kaldi real-time inference is called “online decoding”**
  – Kaldi run-time inference expects raw 16-bit audio chunks. They recommend transporting audio as:
    • raw audio over TCP/IP
    • via RTP audio packets received and decoded by GStreamer

• **Kaldi needs wideband audio**
  – for accuracy benchmarks
  – training augmentation, R&D work, published results based on wideband audio

• **GStreamer is weak in telecom / wideband audio support**
  – no support for EVS, AMR-WB support is weak for concurrent threads, reliability
  – lack of advanced handling for packet loss, stream alignment between multiple streams within a call, stream gaps (call waiting, music on hold), etc
  – no support for RFC8108 (multiple streams from one endpoint)
Kaldi Interface, Data Flow

- **SigSRF replaces GStreamer**
  - minimum REST APIs required – session create/delete/modify
  - session create can be specific or give IP:port and let SigSRF auto-detect codec, bitrate, ptime, etc from RTP data flow
  - packet input via UDP, pcap input for R&D, testing purposes

- **Inferlib**
  - we added an interface library that in turn interfaces to Kaldi run-time libs
Kaldi Interface, Software Architecture

App Developer APIs
- DSCreateSession()
- DSSet/GetSessionInfo()
- DSDeleteSession()

Session Data
- Structs
  - SESSION_DATA
  - TERMINATION_INFO
  - voice_attributes

Packet / Media Threads

Input Pkt Queues
- Stream 0
- Stream 1
- Stream N-1

Output Pkt Queues
- Jitter Buffer 0..N-1
- Transcoded 0..N-1
- Stream Group 0..M-1

Packet Loss Mitigation

Stream Groups
- audio / video domain
- multiple streams

Stream Groups

FLC Signal Processing
- audio processing
- output pkt format

Interlib
- supports multiple models (ASR, SID, etc)
- real-time, concurrent threads

Kaldi Libs
- C/C++ shared libs
- modules used by Kaldi "recipes"
  (scripts + executables)

Libs

Packet Loss Mitigation

Jitter Buffer
- Add
- Get

Jitter Buffer

Decode

Encode

FS Conv

Inferlib

SigSRF Libs
- Kaldi Libs
- Open Source

USB Audio

wav, other audio format files

ASR text

wav files

pcap files

Libs

Packet / Media Threads

Kaldi ASR Demo, Analytics Mode
© Signalogic 2019-2020
Rev 2, Aug 2020

Signalogic, not under NDA
KubeEdge Integration

- **SigSRF and Kaldi libs inside KubeEdge container**
  - minimum 4 x86 cores, 32 GB mem, 1 TB HDD

- **Mobile device app**
  - creates ASR sessions with REST APIs
  - push-to-talk, send codec output packets via UDP/RTP
  - possibly we can send copies of in-call codec packets
**Kaldi Info: Run-Time Inference**

- **One end-to-end thread on one Xeon x86 core**
  - input is 16-bit raw audio, output is ASR text (plus logs, stat files, etc)
  - they maintain a sweet spot of about 2x RTF (real-time factor). They don’t use OpenMP, TBB, or other HPC multicore methods
  - ARM cores can be used, but support on Kaldi user groups is limited
- **Main Kaldi contributors are focused on state-of-the-art R&D**
  - DNN and HMM architecture, improved training are priorities, not performance
  - not focused on concurrent streams, high capacity, reliability, etc.
Kaldi Info: Integration

• **Kaldi is its own framework**
  – main Kaldi contributors are working on PyTorch support
  – partially supports TensorFlow, but main contributors no longer working on it
  – no support for Caffe, MXnet, etc.

• **To integrate Kaldi into production applications takes effort**
  – developer interface is based on Linux shell scripts, so we tracked inference scripts + binaries to find necessary APIs that inferlib must support
  – if you ask questions on kaldi-asr.org about improving performance, reducing model size, concurrent threads, etc you will get general advice only

• **Acceleration**
  – GPUs are supported by Nvidia tech personnel on kaldi-asr.org
  – also seems to be the case for OpenVINO (Intel)
Kaldi Info: Architecture, DNNs

• Architecture
  – uses “chain” models: DNN¹ + xMM²
  – AM (acoustic model) recognizes phonemes
  – phonemes vary depending on context, so “tri-phones” are used
  – LM (language model) recognizes words as tri-phone combinations

• DNN frequency domain data
  – formed by sliding FFT analysis of incoming time series data. Each FFT frame output is similar to cochlea in human ears
  – groups of FFT frames form images
  – successive images are called “TDNN” (time delayed DNN), similar to series of CNNs³

• Training
  – DNNs saved as “x-vectors” and “i-vectors”
  – HMM / GMMs saved as FSTs⁴

¹ Deep Neural Network, ² Hidden Markov Model, Gaussian Mixed Model, ³ Convolutional Neural Network, ⁴ Finite State Transducer