Compressed Offload

with

PulseAudio & GStreamer

Arun Raghavan
Ford_Prefect | @louiswu

“Modern” audio hardware

```
+---------+      +---------+      +---------+
|         |---->|         |---->|         |---->🔈
|   CPU   |     |   DSP   |     |  CODEC  |
|         |<----|         |<----|         |<----🎙
+---------+      +---------+      +---------+
```

Processing

Flexibility

Power savings

CPU sends encoded data

goes to sleep

DSP does decode + render

Sounds simple enough

- Detect and expose formats
- Allow clients to negotiate
- Stream audio data (frames)
- Smug satisfaction of watts saved

Our kryptonite is the past

Everything is PCM (ish)

Bytes ≈ Time

1920 / S16LE / 2ch / 48 kHz ≈ 10 ms

DSP decoders can be finicky

Precise parameters at init

Less tolerant of “bad” streams

ALSA `compress_offload`

- Query “caps”
- Set parameters
- Write data
- Get timestamp

PulseAudio

API

- Extended API: `pa_format` ≈ `GstCaps`
- Sink can expose supported formats
- Client can propose a list of formats
- Core selects one and tells client

API

- Protocol and stream API are bytes-based
- Data is written in arbitrary byte chunks
- Latency and timing based on buffer sizes (bytes)

Server

- Deals with a stream, not tracks
- Renders silence when there is no data
- Does mixing, conversion, volumes
- Rewinds

- Add a bunch of new formats for MP3/AAC/…
- Disallow arbitrary buffer position writes
- Assume each buffer written is one frame

- Modify the protocol for timestamp & duration
- Add per-buffer flags in protocol (discont)

- Add a API to set the format on a sink
- Allow sinks to not render data on IDLE
- Add API to flush & drain on sinks
- Disable rewinding on compressed streams

- No public sink implementation yet
- Compress offload sink Should Be Easy™
- Not much hardware (DragonBoard?)

GStreamer

- `pulsesink` based on `GstAudioBaseSink`
- Which needs a `GstAudioRingBuffer`
- Which works with bytes/samples
- Changing these requires radical surgery

- Bypass the problem with `pulsedirectsink`
- No ringbuffer — just write buffers as they come
- Use flush and drain API for flushing and EOS

- Parsers need to be accurate
- `aacparse` often misses HE-AAC extensions
- Ditto `asfparse` for WMA
- Vorbis & FLAC have `streamheader` in caps

Future

Merge all the work

`compress_offload` sink

Timing and latency

Routing of PCM & compressed

Compressed capture

Gapless playback

References

https://gitlab.freedesktop.org/arun/pulseaudio/commits/compressed

https://gitlab.freedesktop.org/arun/gst-plugins-good/commits/pulsedirectsink

https://www.kernel.org/doc/html/latest/sound/designs/compress-offload.html

Most of the work funded by

Qualcomm

Props to them for helping push this forward

Questions?

♥