

# The Story of PulseAudio and Compressed Offload

Arun Raghavan  
Ford\_Prefect | @louiswu

# The A Linux Audio Stack

```
+-----+  
| Application |  
+-----+
```

↓

```
+-----+  
| GStreamer |  
+-----+
```

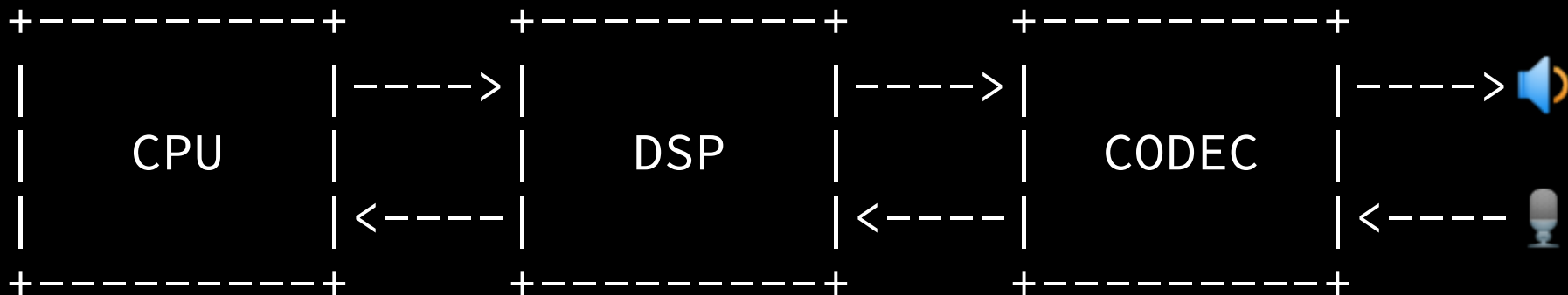
↓

```
+-----+  
| PulseAudio |  
+-----+
```

↓

```
+-----+  
| ALSA |  
+-----+
```

“Modern” audio hardware



Processing

Flexibility

Power savings

# *Compressed Offload*

CPU sends encoded data

Goes to sleep

DSP does decode + render

```
+-----+
| Application |
+-----+
```

↓

mp3

```
+-----+
| GStreamer |
+-----+
```

↓

pcm

```
+-----+
| PulseAudio |
+-----+
```

↓

pcm

```
+-----+
| ALSA |
+-----+
```



```
+-----+
| Application |
+-----+
```

↓

mp3

```
+-----+
| GStreamer |
+-----+
```

↓

mp3

```
+-----+
| PulseAudio |
+-----+
```

↓

mp3

```
+-----+
| ALSA |
+-----+
```

*Sounds* simple enough

Detect and expose formats

Allow apps to negotiate

Stream audio data (frames)

Smug satisfaction of watts saved

Our kryptonite is the past

Everything is PCM (ish)

Bytes  $\approx$  Time

1920 / S16LE / 2ch / 48 kHz  $\approx$  10 ms

Not true for compressed audio

ALSA compress\_offload

Query capabilities

Set parameters

Write data

Get timestamp

PulseAudio: Clients

**pa\_format\_info**: Flexible key/value pairs

Sink can expose supported formats

Client can propose a list of formats

Core selects one and tells client



Protocol and stream API are bytes-based

Data is written in arbitrary byte chunks

Latency and timing based on buffer sizes (bytes)

PulseAudio: Sinks

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

Add API to flush & drain on sinks

Allow sinks to not render data on IDLE

Don't rewind compressed streams

No upstream sink implementation yet

Compress offload sink Should Be Easy™

Not much hardware (DragonBoard?)

GStreamer



`pulsesink` element

Uses `GstAudio` base-classes

Works with bytes/samples

Changing this requires radical surgery

`pulsedirectsink` element

Bypass the problem

No ringbuffer

Just write buffers as they come

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

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?

