Chromium OS audio

CRAS audio server
Why another audio server?

- low end hardware (1 core atom, or Tegra 2)
- optimize for one user (chrome)
- dynamic stream re-routing
- maintainability, code size, security
Basic Audio Flow

- Chromium
  - dbus control
  - Playback/Capture shm
  - CRAS
    - BlueZ
    - ALSA
Client Library API

```c
int cras_client_create(struct cras_client **client);

int cras_client_run_thread(struct cras_client *client);

struct cras_stream_params *cras_client_unified_params_create(
    enum CRAS_STREAM_DIRECTION direction, /* direction - CRAS_STREAM_OUTPUT or CRAS_STREAM_INPUT */
    unsigned int block_size, /* block_size - The number of frames per callback(dictates latency). */
    enum CRAS_STREAM_TYPE stream_type, /* not currently used */
    uint32_t flags, /* not used either. */
    void *user_data, /* user_data - Pointer that will be passed to the callback. */
    cras_unified_cb_t unified_cb, /* unified_cb - Called for each block size samples */
    cras_error_cb_t err_cb, /* err_cb - Called when there is an error with the stream. */
    struct cras_audio_format *format); /* format - Specifies bits per sample, num chan, sample rate */

int cras_client_add_stream(struct cras_client *client, cras_stream_id_t *stream_id_out, struct cras_stream_params *config);
```
Server side features

Timer Based Wake-ups based on stream level
Device Sample Rate Estimation
Mixing, DSP, and format conversion
Volume level tuning
Device synchronization
One audio thread
Wake up timing

Wakes up each stream based on a timer
Timer rate set based on block size
Adjusted based on estimated device clock
Underrun handling
Don’t let one stream cause all to glitch
Scheduling jitter, real time threads help
Timing picture

One 44.1kHz stream (good)
Timing glitch

Interactive governor preempts
Device Rate Estimation

By measuring the buffer level at known times estimate the sample rate of the device.

Use this estimated sample rate to calculate how long each stream should sleep.
Avoid copying audio

Shared memory between client and server

Server moves audio directly from shm to mmapped device buffer

Format conversion still needs an extra copy.
Buffer write/read point management is tricky

n devices reading from the same stream and n streams writing to the same device
Write pointer management

Each device tracks offset of all streams attached to it.

Similarly each stream tracks its offset into each device it is attached to.

offsets are updated after all users have mixed
Two output stream example

- HW level 100
  - S1: 40
  - S2: 0

- HW level 100
  - S1: 60
  - S2: 0

- HW level 100
  - S1: 60
  - S2: 20

- HW level 120
  - S1: 40
  - S2: 0
Output Processing
Output processing

Need Speaker EQ

Each system is different

Different OEMs want different tunings
DSP speed

Output and input processing are on the critical path.

Heavily optimized with neon and sse versions.

Keep it simple with a three band DRC and a 10 band eq per channel.
DSP tuning

Can listen in real time on un-tuned device, WebAudio blocks are equivalent to the blocks used in the optimized DSP.

Generated config file, hands off tuning for core engineering team.
ALSA UCM usage

Use a single ‘HiFi’ verb.

Have devices for headphones, external microphones, and HDMI.

A few non-standard values.
SectionVerb {
  Value {
    OutputDspName "speaker_eq"
  }
  EnableSequence [
    cdev "hw:Venice2"
    cset "name='Left Speaker Mixer Left DAC Switch' on"
    cset "name='Right Speaker Mixer Right DAC Switch' on"
    ...snip...
    cset "name='Headphone Left Switch' on"
    cset "name='Headphone Right Switch' on"
    cset "name='Speaker Left Switch' on"
    cset "name='Speaker Right Switch' on"
    cset "name='Speakers Switch' on"
  ]
  DisableSequence [
  ]
}
SectionDevice."Headphone".0 {
  Value {
    JackName "NVIDIA Tegra Venice2 Headphone Jack"
    OutputDspName ""
  }
  EnableSequence [
    cdev "hw:Venice2"
    cset "name='Speakers Switch' off"
    cset "name='HP Left Out Switch' on"
    cset "name='HP Right Out Switch' on"
  ]
  DisableSequence [
    cdev "hw:Venice2"
    cset "name='HP Left Out Switch' off"
    cset "name='HP Right Out Switch' off"
    cset "name='Speakers Switch' on"
  ]
}
SectionDevice."Mic".0 {
    Value {
        JackName "NVIDIA Tegra Venice2 Mic Jack"
        CaptureControl "MIC2"
    }
    EnableSequence [
        cdev "hw:Venice2"
        cset "name='Int Mic Switch' off"
        cset "name='DMIC Mux' ADC"
        cset "name='Mic Jack Switch' on"
    ]
    DisableSequence [
        cdev "hw:Venice2"
        cset "name='Mic Jack Switch' off"
        cset "name='DMIC Mux' DMIC"
        cset "name='Int Mic Switch' on"
    ]
}
External Device Support
USB/Bluetooth

Bluetooth chip attached through USB not i2s

USB devices go through ALSA

Bluetooth through a Bluez created socket A2DP/HFP/HSP
USB/Bluetooth transfer size

Main challenge is granularity of transfers

Data is sent over USB in URB sized chunks
For Bluetooth one MTU can be > 500 samples
No way to detect the size from user space
Have to pad buffers to ensure enough is ready
HDMI audio output

Auto routing decision is difficult

EDID parsing helps

Docked mode
Improvements for embedded systems

Process hop eliminated on one user systems

Add local streams

Make timing and device management a separate library
Avoid waking up

Could improve a lot here

Don’t try to synchronize streams at all

Favor accuracy of stream callbacks over wakeup aggregation.
CPU usage

measure # of instructions over 5 seconds of playback, averaged three runs each.

perf stat -p <server pid>,<player pid> -r 3 -a sleep 5

All tests were on a TegraK1 chromebook. Crouton was used to run pulseaudio
localhost / # perf stat -p 12912,12922 -r 3 -a sleep 5

Performance counter stats for process id '12912,12922' (3 runs):

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Rate</th>
<th>Utilization</th>
<th>Max/Min Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-clock (msec)</td>
<td>940.900007</td>
<td># 0.188 CPUs utilized</td>
<td>( +- 26.80%)</td>
<td>[100.00%]</td>
</tr>
<tr>
<td>context-switches</td>
<td>4,937</td>
<td># 0.005 M/sec</td>
<td>( +- 1.18%)</td>
<td>[99.98%]</td>
</tr>
<tr>
<td>cpu-migrations</td>
<td>6</td>
<td># 0.007 K/sec</td>
<td>( +- 92.22%)</td>
<td></td>
</tr>
<tr>
<td>page-faults</td>
<td>0</td>
<td># 0.000 K/sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cycles</td>
<td>398,970,008</td>
<td># 0.424 GHz</td>
<td>( +- 10.34%)</td>
<td></td>
</tr>
<tr>
<td>instructions</td>
<td>85,783,097</td>
<td># 0.22 insns per cycle</td>
<td>( +- 6.28%)</td>
<td></td>
</tr>
<tr>
<td>branches</td>
<td>21,426,180</td>
<td># 22.772 M/sec</td>
<td>( +- 9.23%)</td>
<td></td>
</tr>
<tr>
<td>branch-misses</td>
<td>2,620,234</td>
<td># 12.23% of all branches</td>
<td>( +- 0.85%)</td>
<td></td>
</tr>
<tr>
<td>5.005028610 seconds time elapsed</td>
<td></td>
<td></td>
<td>( +- 0.06%)</td>
<td></td>
</tr>
</tbody>
</table>
Single 44.1kHz wav file with aplay

`aplay -D<plugin> -B20000 filename.wav`

<table>
<thead>
<tr>
<th>plugin</th>
<th>instructions (millions)</th>
<th>task-clock (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>hw:1</td>
<td>11.2</td>
<td>294</td>
</tr>
<tr>
<td>pulse</td>
<td>188.9</td>
<td>1686.4</td>
</tr>
<tr>
<td>cras</td>
<td>85.8</td>
<td>940.9</td>
</tr>
</tbody>
</table>
One 44.1kHz one 48kHz

```bash
aplay -D<plugin> -B20000 44_1k.wav
aplay -D<plugin> -B20000 48k.wav
```

<table>
<thead>
<tr>
<th>plugin</th>
<th>instructions (millions)</th>
<th>task-clock (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pulse</td>
<td>576.2</td>
<td>1755.1</td>
</tr>
<tr>
<td>cras</td>
<td>247.1</td>
<td>837.8</td>
</tr>
</tbody>
</table>
Native clients at 44.1kHz

pacat --rate 44100 --latency 1764 --raw /dev/zero
ctc --playback_file /dev/zero --block_size 441 --rate 44100

<table>
<thead>
<tr>
<th></th>
<th>85.1</th>
<th>1506.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>pacat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ctc</td>
<td>27.4</td>
<td>673.5</td>
</tr>
</tbody>
</table>
## Native clients 44.1kHz and 48kHz

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pacat</td>
<td>356.1</td>
<td>1613.0</td>
</tr>
<tr>
<td>ctc</td>
<td>102.0</td>
<td>1506.0</td>
</tr>
</tbody>
</table>