Pipewire Audio Backend in QEMU

Dorinda Bassey
Software Engineer
dbassey@redhat.com

Red Hat

What is it?

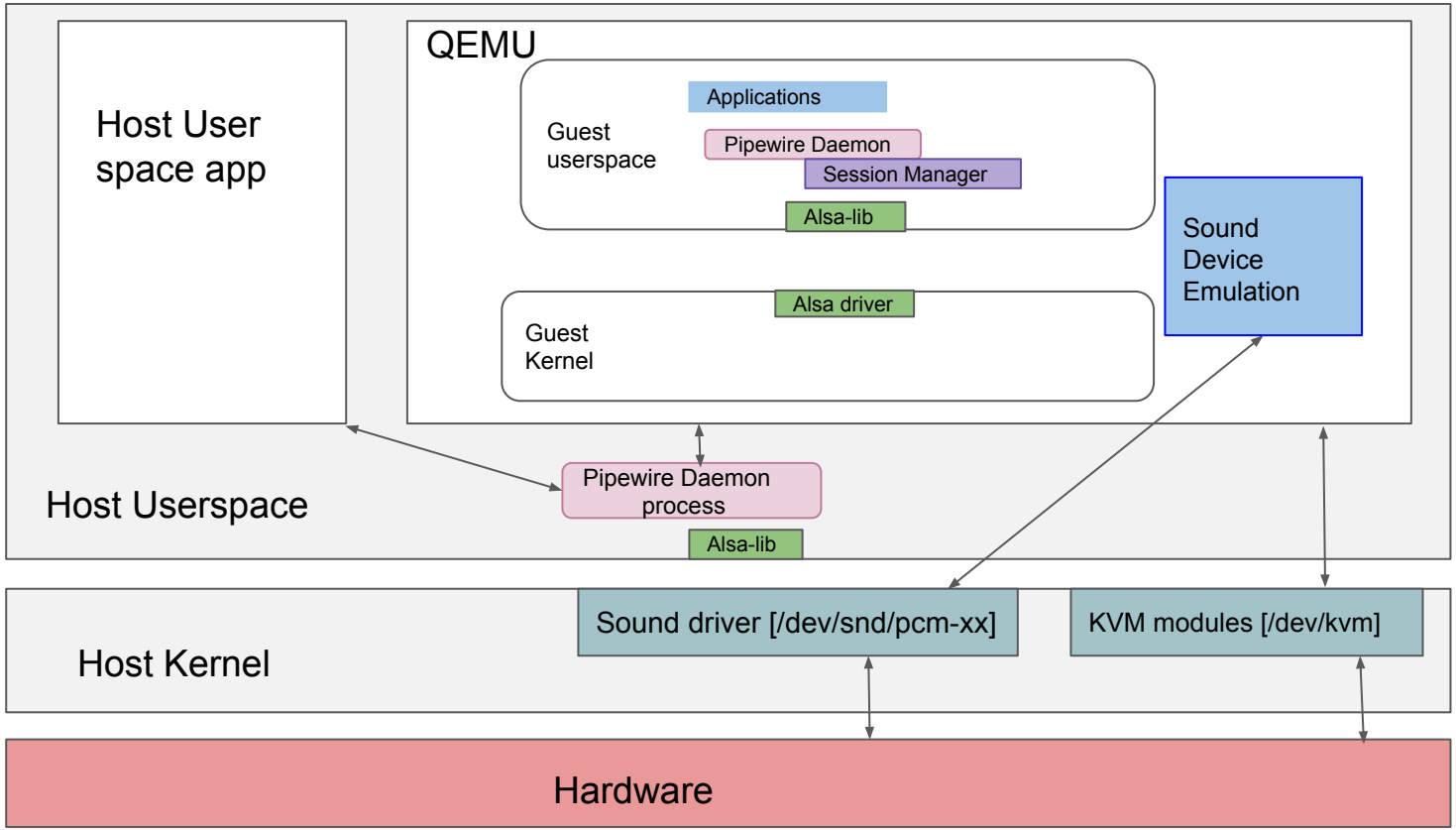
Qemu Audio backend

- component responsible for managing audio streams
- handles audio input and output for virtual machines running on QEMU

Pipewire Audio Backend

- provide native pipewire support for QEMU using pipewire C APIs and libraries.

Architectural Overview



When Upstreamed?

- Patch was merged in May 2023.
  - Pipewire audio backend feature was added in QEMU 8.1 release
- Supports pipewire version >=0.3.60

Audio Backends

```
$ qemu-system-x86_64 -audiodev help
```

Available audio drivers:

- none
- alsa
- dbus
- jack
- oss
- pa
- pipewire
- sdl
- spice
- wav

Pipewire backend-specific Parameters

```
$ qemu-system-x86_64 -audiodev pipewire,id=pwsnd,\
out.name=source,out.stream-name=pw,out.latency=15000,
```

QAPI Schema:

```
{ 'struct': 'AudiodevPipewirePerDirectionOptions',
  'base': 'AudiodevPerDirectionOptions',
  'data': {
        '*name': 'str',
        '*stream-name': 'str',
        '*latency': 'uint32' }
}
{ 'struct': 'AudiodevPipewireOptions',
  'data': {
        '*in':   'AudiodevPipewirePerDirectionOptions',
        '*out': 'AudiodevPipewirePerDirectionOptions' } }
```

Generic audio backend Parameter

```
$ qemu-system-x86_64 -audiodev pipewire,id=pwsnd,out.latency=15000,\
out.channels=2,out.mixing-engine=true,out.buffer-length=10000,out.format=s1
6
```

QAPI Schema:

```
{ 'struct': 'AudiodevPerDirectionOptions',
  'data': {
            '*mixing-engine':  'bool',
            '*fixed-settings': 'bool',
            '*frequency':          'uint32',
            '*channels':           'uint32',
            '*voices':       'uint32',
            '*format':       'AudioFormat',
            '*buffer-length':  'uint32' } }
```

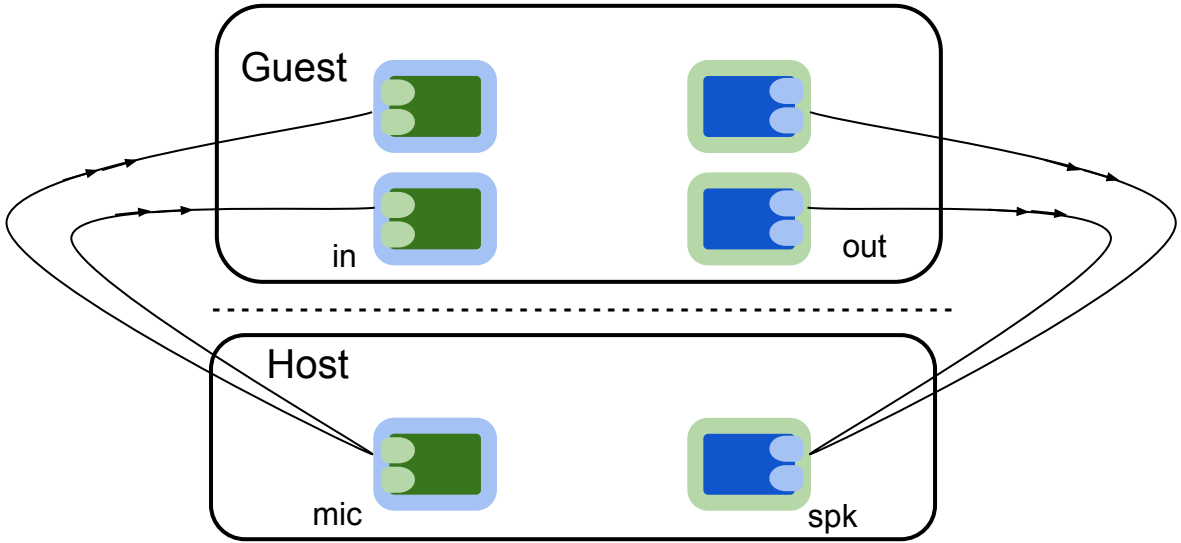
Audio device Example

```
$ qemu-system-x86_64 -audiodev pipewire,id=pwsnd \
-device intel-hda -device hda-duplex,audiodev=pwsnd
```

```
#define DEFINE_AUDIO_PROPERTIES(_s, _f)          \
        DEFINE_PROP_AUDIODEV("audiodev", _s, _f)
#endif /* QEMU_AUDIO_H */

static Property hda_audio_properties[] = {
        DEFINE_AUDIO_PROPERTIES(HDAAudioState, card),
        DEFINE_PROP_UINT32("debug", HDAAudioState, debug,   0),
        DEFINE_PROP_BOOL("mixer", HDAAudioState, mixer,  true),
        DEFINE_PROP_BOOL("use-timer", HDAAudioState, use_timer,  true),
        DEFINE_PROP_END_OF_LIST(),
};
```

Multiple audio backends with Pipewire

```
$ qemu-system-x86_64 \
  -audiodev pipewire,id=pwsnd0,out.channels=1 \
  -audiodev pipewire,id=pwsnd1,out.latency=15000,out.channels=2,in.channels=2 \
  -device intel-hda -device hda-output,audiodev=pwsnd0 \
  -device intel-hda -device hda-duplex,audiodev=pwsnd2
```



Red Hat

Playback with Pipewire Backend

Activate stream for playback

pw_stream_set_active()

qpw_buffer_get_free()

pw_thread_loop_lock()

Playback with Pipewire Backend

Activate stream for playback

pw_stream_set_active()

qpw_buffer_get_free()

pw_thread_loop_lock()

If (PW_STREAM_STATE_STREAMING);

spa_ringbuffer_get_write_index()

If (filled < 0);

Report buffer underrun;

else

Report overrun write;

get Avail no of bytes

If (len > avail);

len = avail;

spa_ringbuffer_write_data(pw_buffer, indx, data)

Index += len;

spa_ringbuffer_write_update(index)

else

pw_thread_loop_unlock()

Playback with Pipewire Backend

Activate stream for playback

pw_stream_set_active()

qpw_buffer_get_free()

pw_thread_loop_lock()

If (PW_STREAM_STATE_STREAMING);

spa_ringbuffer_get_write_index()

get Avail no of bytes

spa_ringbuffer_write_data(pw_buffer, indx, data)

Index += len;

spa_ringbuffer_write_update(index)

else

pw_thread_loop_unlock()

If (filled < 0);

Report buffer underrun;

else

Report overrun write;

If (len > avail);

len = avail;

Data processing to Read from buffer

b = pw_stream_dequeue_buffer()

req = b->requested * frame_size;

n_byts = MIN(req, b->data.maxsize);

avail = spa_ringbuffer_get_read_index()

If (b == NULL);

Err (out of buffers);

If (avail <= 0);

audio_pcm_info_clear_buf();

else

If (avail < n_bytes);

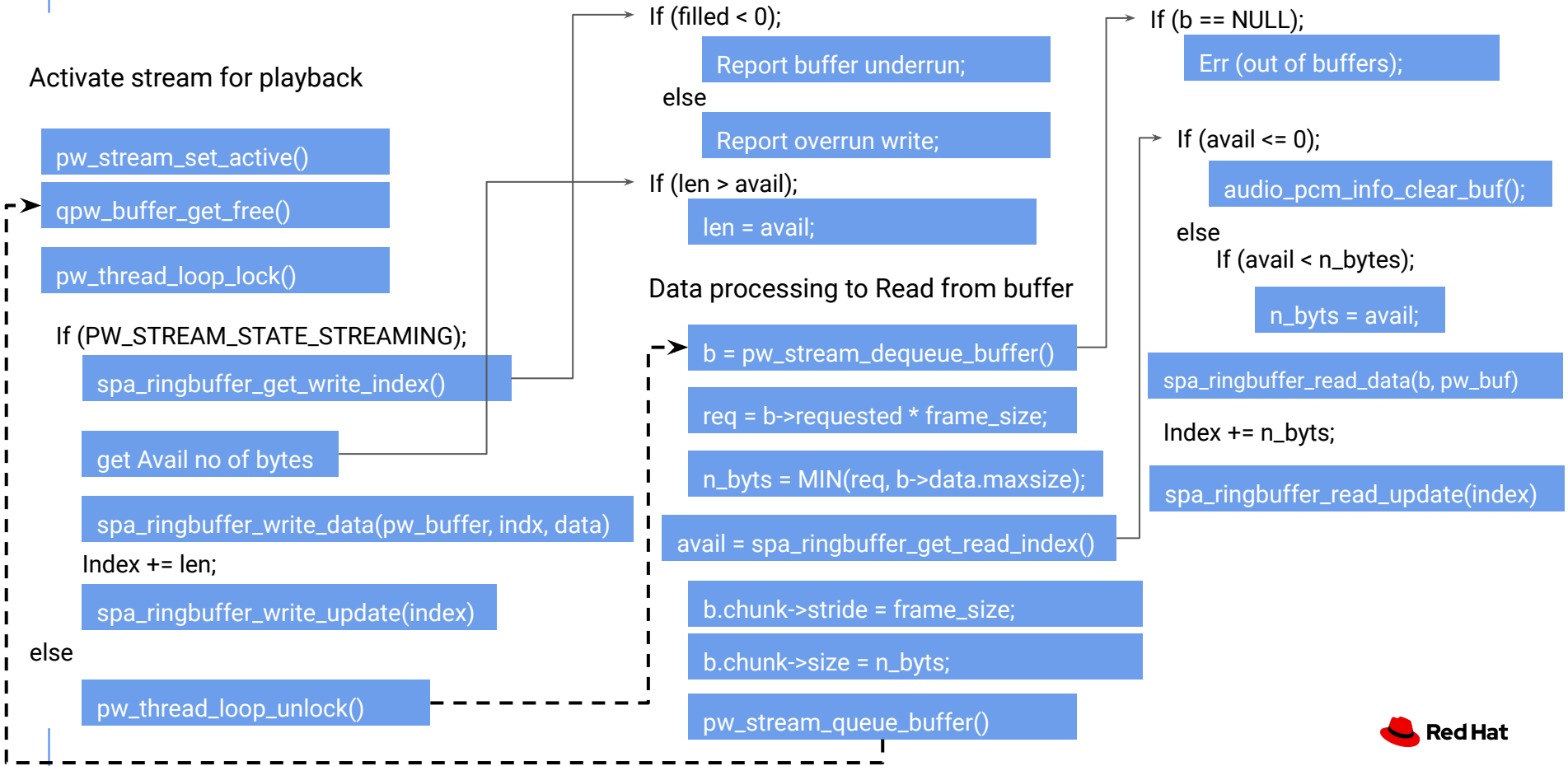
n_byts = avail;

spa_ringbuffer_read_data(b, pw_buf)

Index += n_byts;

spa_ringbuffer_read_update(index)

Playback with Pipewire Backend



Red Hat

Capture with Pipewire Backend

Activate stream for capture

pw_stream_set_active()

qpw_buffer_get_free()

pw_thread_loop_lock()

Capture with Pipewire Backend

Activate stream for capture

pw_stream_set_active()

qpw_buffer_get_free()

pw_thread_loop_lock()

If (PW_STREAM_STATE_STREAMING);

avail = spa_ringbuffer_get_read_index()

get Avail no of bytes

If (len > avail);

len = avail;

spa_ringbuffer_read_data(pw_buffer, indx, data)

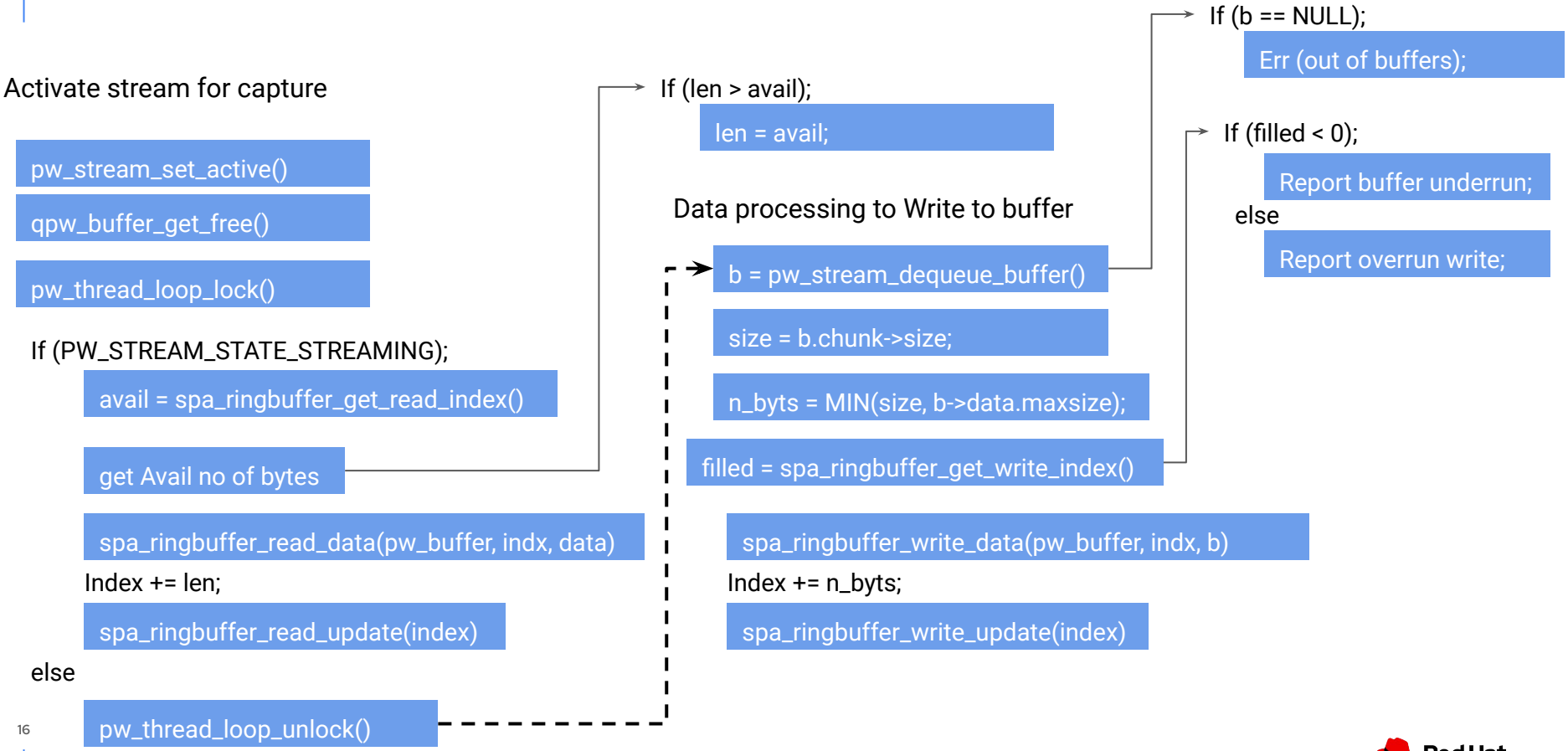
Index += len;

spa_ringbuffer_read_update(index)

else

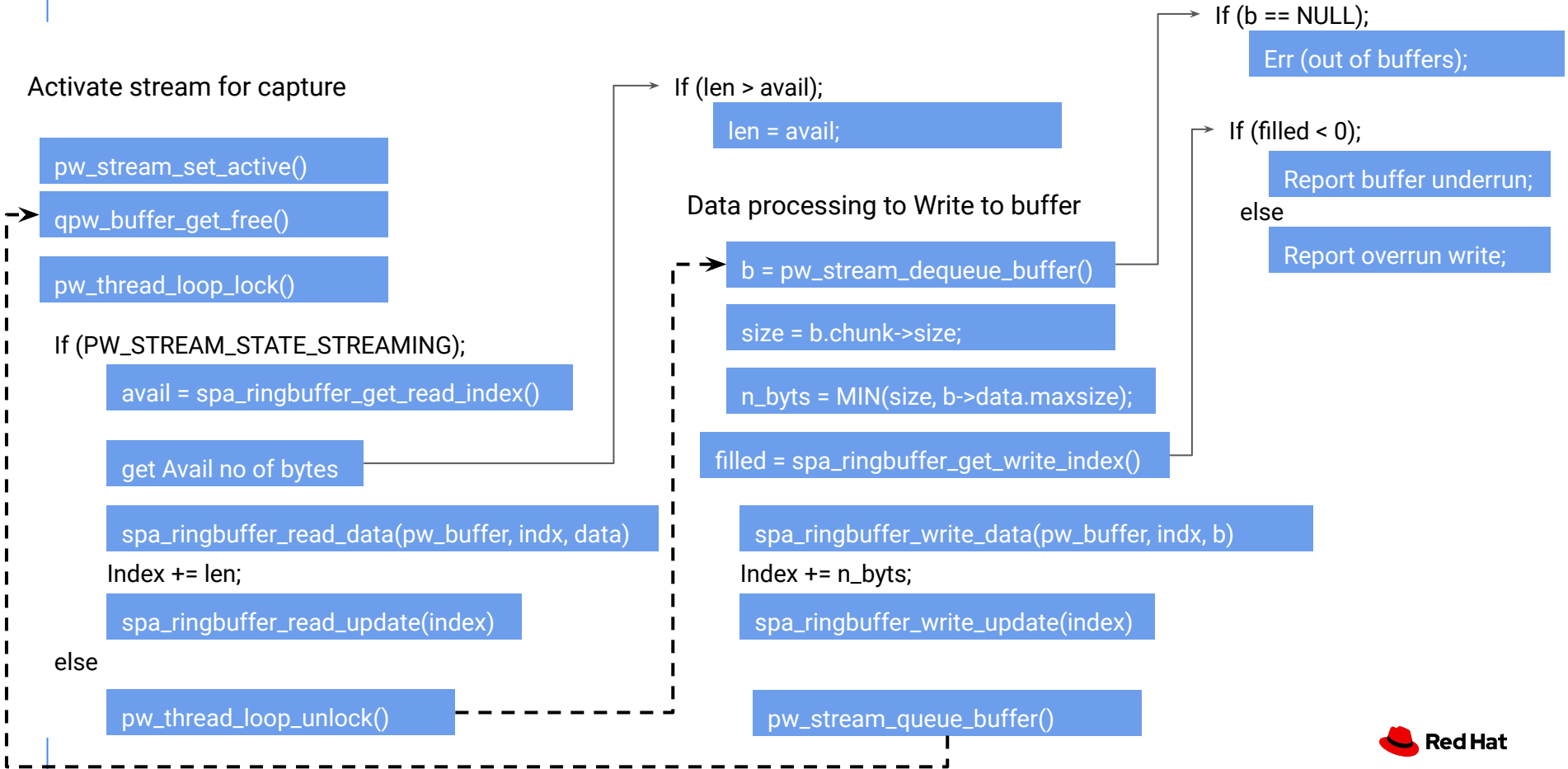
pw_thread_loop_unlock()

Capture with Pipewire Backend



Red Hat

Capture with Pipewire Backend



Red Hat

Volume Controls

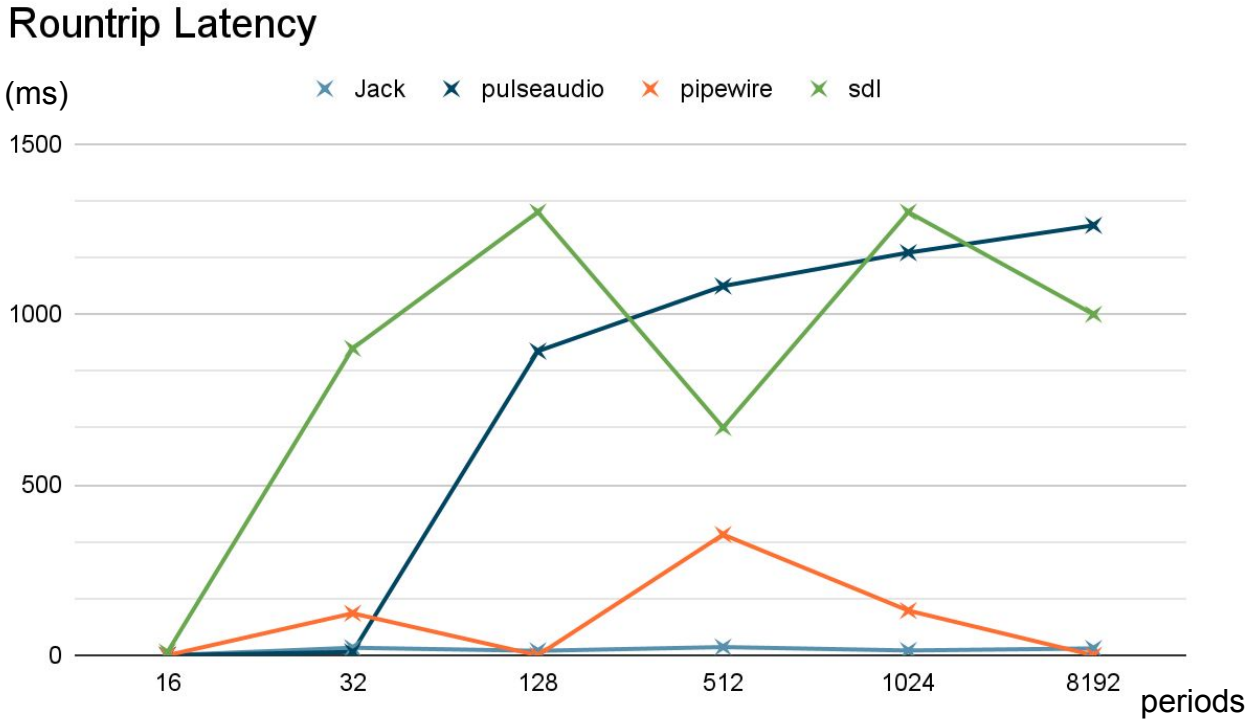
- Allow for control of volumes on the guest to be synchronize with volumes on the host.
- 0.0 is silence, 1.0 is without attenuation.
  - This is the effective volume that is applied.

```
pw_thread_loop_lock();

for (i = 0; i < vol->channels; ++i) {
        pw->volume.values[i] = (float)vol->vol[i] / 255;
}

pw_stream_set_control(SPA_PROP_channelVolumes,
v->volume.values);

pw_thread_loop_unlock();
```

Pipewire Backend Features

- Low latency
  - Significantly reduced latency both in playback/capture
  - Can dynamically change latency
- Reduced footprint/dependencies over the current audio backends
- Better integration between audio applications running on the guest and communication with the host.
- Native pipewire API - benefits from pipewire less overhead (less cpu usage and memory)

Roundtrip latency benchmarking



Red Hat

Debugging

- QEMU supports working with gdb
- QEMU internal tracing infrastructure:
  - Trace events - can be configured on the command line
  - E.g: `---trace pw_vol` `--trace pw_write` `--trace pw_period`
- PIPEWIRE_DEBUG logging:
  - Enable debugging categories using the PIPEWIRE_DEBUG environment variable
  - E.g: PIPEWIRE_DEBUG=3

Helpful links

https://medium.com/@dorindabassey/pipewire-audio-backend-in-qemu-be014359475

https://www.kraxel.org/blog/2020/01/qemu-sound-audiodev/

https://www.qemu.org/docs/master/system/invocation.html#hxtool-0

https://gitlab.freedesktop.org/pipewire/pipewire/-/wikis/Performance#

Questions

Thank you