Implementing virtio-sound in rust-vmm project

Dorinda Bassey  
dbassey@redhat.com

Matias Vara Larsen  
mvaralar@redhat.com
Outline

- Automotive use case
- Protocol Overview
- Virtio-sound device and driver
- Vhost-user implementation
- Audio backend architecture
- Upstream status
- Questions
Use case: Automotive

Host userspace -> Vhost-user sound Device

VMM

Guest userspace

Apps

Guest kernel

Qemu

Android Applications

Android Audio framework

Android HAL services

Virto-sound driver

Guest Kernel

Sound Driver

Host Kernelspace

KVM

https://source.android.com/docs/automotive/virtualization/reference_platform
What is virtio-sound?

- Paravirtualized sound device
  - VIRTIO spec 1.2: [5.14 Sound Device](#)

Consisting of:

- Virtio-sound driver
- PCI transport
- Vhost-user sound device

![Diagram of virtio-sound architecture](image-url)
Overview of the Protocol

Vhost-user protocol

- Unix Domain socket for the control plane
- Consist of
  - Frontend sending message request
  - Backend sending message replies
- Establish Virtqueues sharing between the Host application and the virtio-sound driver in the Guest
Example of vhost-user protocol message

Dump unix domain socket traffic [1]

$ sudo ./sockdump.py --format hexstring /tmp/vhost4.socket

The size of the Payload is indicated in Size and depends on the REQ

Accessing guest’s memory through vhost-user protocol

Guest

Front-end
Virtio-sound driver

Guest Memory

mmap(fd)

Host

Back-end
Virtio-sound device

Sends “fd”

Sends VQ addr

mmap(fd)
Virtio-sound device and driver

Feature Bits Negotiation:

- At device initialization
- set feature flags based on the feature bits negotiated

```rust
fn features(&self) -> u64 {
    1 << VIRTIO_F_VERSION_1
    | 1 << VIRTIO_F_NOTIFY_ON_EMPTY
    | 1 << VIRTIO_RING_F_INDIRECT_DESC
    | 1 << VIRTIO_RING_F_EVENT_IDX
    | VhostUserVirtioFeatures::PROTOCOL_FEATURES.bits()
}
```
Virtio-sound device and driver

Virtqueues: a mechanism for bulk data transport on virtio devices
Consist of 3 parts
- Descriptor Area
- Driver Area
- Device Area
Virtio sound device and driver: How it works?

Virtio-sound Driver:

- creates a descriptor chain
- writes a descriptor index into the available ring
Virtio-sound device and driver

- **Virtio-sound device and driver**
- **VIRTIO_SND_R_PCM_* messages**
- **VIRTIO_SND_S_*
- **Control Queue**
- **Event Queue**
- **TX Queue**
- **RX Queue**

Red Hat
Virtio-sound device and driver

VIRTIO_SND_R_PCM_* messages

Control Queue
Event Queue
TX Queue
RX Queue

VIRTIO_SND_S_*

Virtio-sound driver
Virtio-sound device

For outputting
For inputting
Playback
Capture
Virtio-sound device and driver

VIRTIO_SND_R_PCM_* messages

For outputting

Playback

Capture

VIRTIO_SND_S_*

For inputting
Vhost-user-sound device implementation

Vhost-user sound device (backend)

Vhost-user protocol

Vhost-device

Vhost-user-sound device implementation

Qemu

Applications

Virtio-sound driver

Virtio-sound-pci device (frontend)

Guest userspace

Guest Kernel

Host userspace

Vhost-device

Vhost-user-sound device (backend)

Vhost-user protocol

Sound Driver

KVM

Host Kernel space

Red Hat
Vhost-user-sound device implementation

15

[1] [PATCH v9 00/11] virtio: cleanup vhost-user-generic and reduce c&p + vhost-user-input
Vhost-user-sound device implementation

- Share device implementations between multiple VMMs
- Use Rust as language so owned versus managed memory and better performance and safety
- Run as a separated process independent of qemu
- Less context switches of the guest app than the built-in device [3]

---

[rust-vmm](https://github.com/rust-vmm/vhost-device)

[2] [https://github.com/rust-vmm/vhost-device](https://github.com/rust-vmm/vhost-device)

[3] [https://gist.github.com/MatiasVara/c69a70a1547ecea0044ece43e4ab9e41](https://gist.github.com/MatiasVara/c69a70a1547ecea0044ece43e4ab9e41)
Vhost-user-sound device implementation

1 Thread for all queues
1 Thread per Stream
1 Stream for input
1 Stream for output
Control Messages Handler

Guest
Virtio-sound driver

Control Queue
available ring
#4 #3 #0

used ring
#0

Control Queue Handler
Parser

Audio Backend
Pipewire

e.g., invoke
pw.set_param(stream=1)
Control Messages Handler

Guest

Virtio-sound driver

Control Queue

available ring

#4 #3 #0

used ring

#0 #0

Control Queue Handler

Parser

Audio Backend

Pipewire

Guest is notified immediately after the msg is processed
Transmission Messages Handler (playback)

Guest

Virtio-sound driver

Tx queue

available ring

#3 #2 #1 #0

used ring

#0

Tx queue handler

FIFO Queue Per Stream

#2

#1

#0

Worker Thread per Stream

Pipewire Buffer

points to

pop()
Transmission Messages Handler (playback)

Guest

Virtio-sound driver

Tx queue

available ring

#3 #2 #1 #0

used ring

#0

Tx queue handler

FIFO Queue Per Stream

#2

#1

#0

Worker Thread per Stream

Worker shall pop a whole period to optimize the CPU usage of the worker thread

Pipewire Buffer

pop()
Transmission Messages Handler (playback)

Virtio-sound driver

Tx queue

available ring

used ring

Tx queue handler

FIFO Queue Per Stream

#0

#1

#2

Notification can be sent ONLY after consumption

Worker Thread per Stream

Pipewire Buffer

pop()

notify guest

points to

Guest Virtio-sound driver

driver
Reception Messages Handler (capture)

The only difference with capturing is that worker fills the driver's buffer.
Virtio-sound device Example

How to launch the vhost-user-sound device daemon on the host

```
$ vhost-device-sound --socket /tmp/snd.sock --backend pipewire
```

Available backends: null, pipewire, alsa

The QEMU invocation to create a chardev socket to allow communication over the vhost-user protocol

```
$ qemu-system-x86_64
   -chardev socket,id=vsnd,path=/tmp/snd.sock
   -device vhost-user-snd-pci,chardev=vsnd,id=snd
   -machine YOUR-MACHINE-OPTIONS,memory-backend=mem
   -m 4096
   -object memory-backend-file,id=mem,size=4G,mem-path=/dev/shm,share=on
   ...
```
Some upstream contributions

- Upstreaming patches for using ack() for the virtio-sound driver
- Upstreaming patches for the virtio-sound specification
- Adding descriptor_utils.rs from virtiofsd to virtio-queue crate
- Alex patches to add a generic vhost-user-device to reduce boilerplate code of all other vhost-user-devices in QEMU
- Developments in pipewire-rs crate:
  - Added thread_loop module and implementation
  - Added spa_ringbuffer FFI functions
  - Bug Fixes to ensure compatibility of virtio-sound device with pipewire-rs crate.
Get in touch!

- Get it from https://github.com/rust-vmm/vhost-device
- Find us at rust-vmm slack channel #virtio-sound at https://rust-vmm.slack.com/
- Take part in our Google Summer of code project, which adds an audio backend to GStreamer for the development of virtio-sound (see https://wiki.qemu.org/Internships/ProjectIdeas/GStreamerVhostDeviceSound)

- Contact us directly:
  - dbassey@redhat.com mvaralar@redhatt.com
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHat