VIRTIO INTRODUCTION

Presented by : Dufy TEGUIA

Course content

1. Devices virtualization
2. Virtio specifications
3. Implementation
4. Go through : virtio example
5. vhost
6. TP : Caesar ciphering device

DEVICES VIRTUALIZATION

Device

- Device presentation in linux
  - A device is represented by a file, in the /dev folder
- Another interface to interact with the device is the sysfs filesystem
  - To get some device attribute or modify its behaviour

Device

- There are two main devices cathegories :
  - Block devices : characterized by the fact that they are more useful when large chunks of data are sent at a time. example : hard drives
  - Character devices that are mostly controled through simple and small data or commands are passed to it example : keyboard

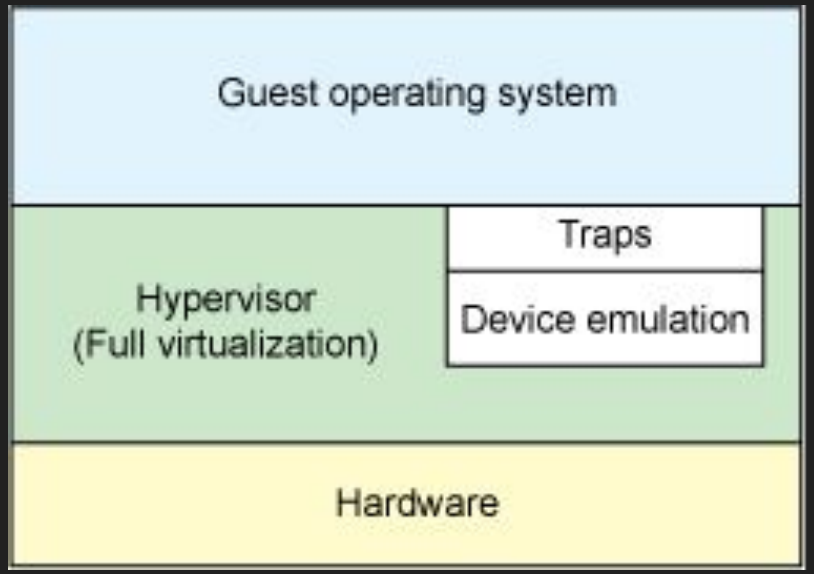
Device

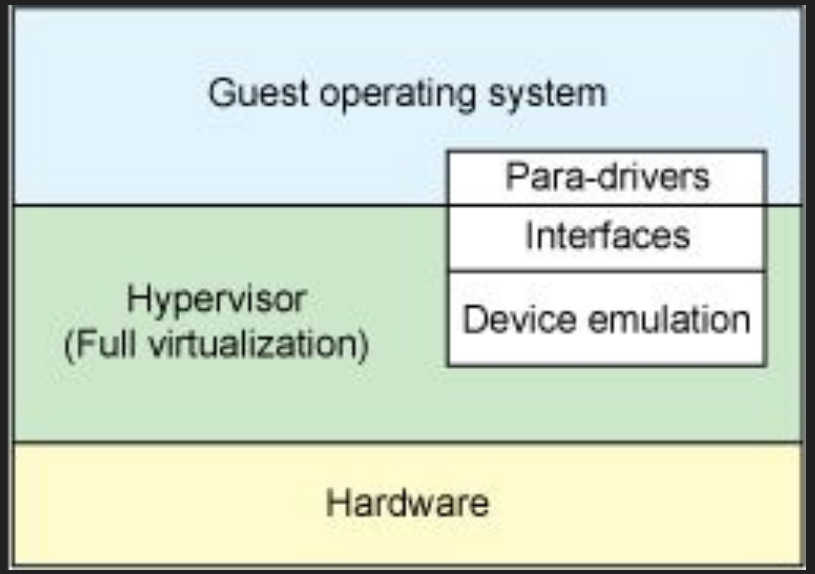
We usually virtualize :

- CPUs
- Memory
- Devices

Device

- The most used virtualization technique nowadays is paravirtualization (Opposed to full virtualization)





Device

- The most used virtualization technique nowadays is paravirtualization (Opposed to full virtualization)

- The de-facto standard for device paramvirtualization is called virtio

VIRTIO SPECIFICATIONS

Virtio Specification

An open specification for virtual machines data I/O communication.

Main idea: make the guest think that it controls real devices.

What we need to ensure is that:

1. The guest is aware of the device
2. The guest can communicate with the device
3. THere is Driver unification
4. Device probing and configuration

Virtio Specification: Expose

How does the host expose the device to VMs?

In real world, PCI hardware expose its configuration space using a specific memory address range

The hypervisor can emulate the same thing to the VM to implement virtio

It is hypervisor specific !

Virtio Specification: Communication

Virtqueue: the mechanism to transfer data between VM and hardware

1. Each device can have multiple virtqueue
2. One virtqueue consist of guest-allocated buffers that the host can interact with

It defines two notifications:

1. Available Buffer Notification:

   Used by the driver to signal there are buffers that are ready to be processed by the device

2. Used Buffer Notification:

   Used by the device to signal that it has finished processing some buffers.

Virtio Specification: Summary

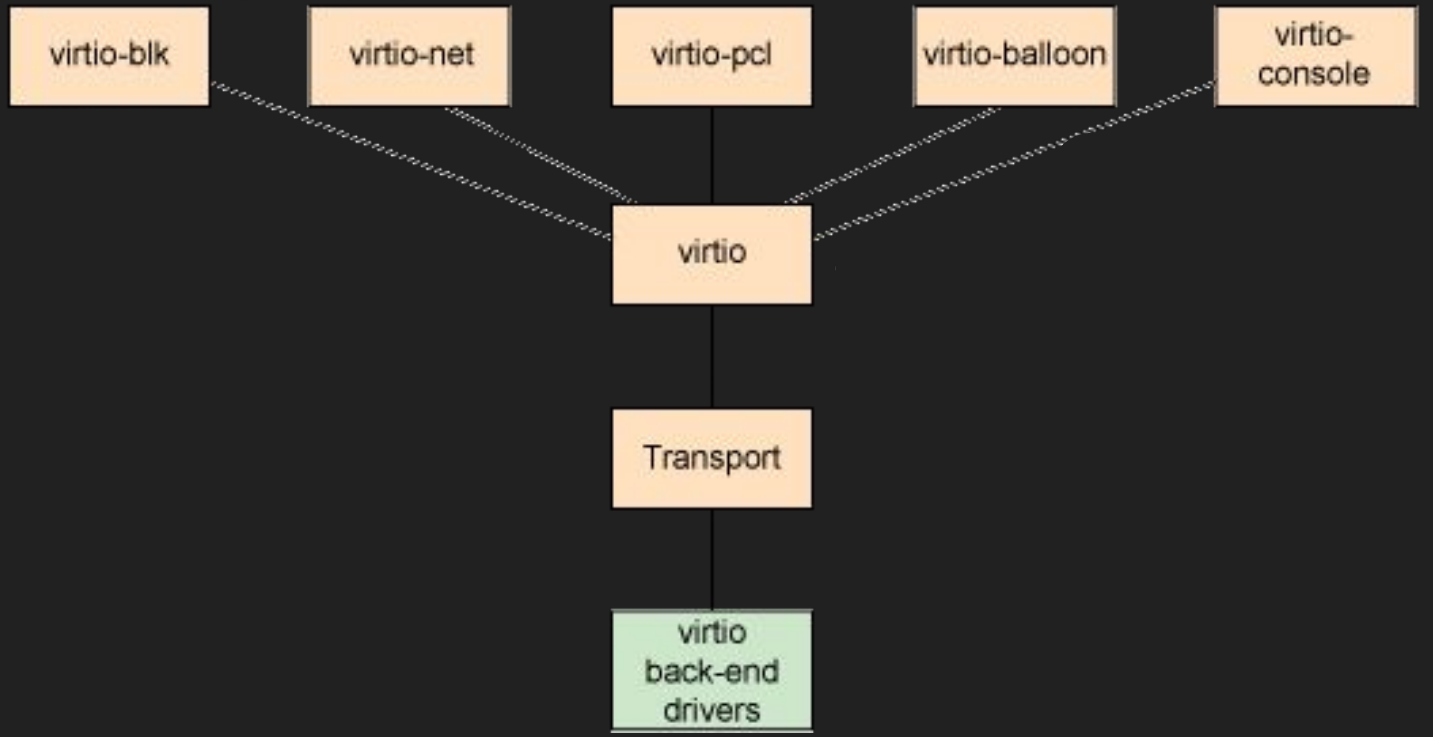
In summary, the virtio driver needs to expose the following to guest

- Device's feature bits
- Status bits
- Configuration space
- Nofications system
- Virtqueues
- Device's transport interface

Documentation :
https://docs.oasis-open.org/virtio/virtio/v1.1/csprd01/virtio-v1.1-csprd01.html

IMPLEMENTATION

VirtIO Architecture



VirtIO Transport layer - Virtqueue

- It is a part of the memory of the guest OS
- A channel between front-end and back-end
- It is an interface Implemented as Vring (a ring buffer) Vring is a memory mapped region between the hypervisor and guest OS
- Vring is the memory layout of the virtqueue abstraction

```
struct virtqueue_ops {
    int (*add_buf)(struct virtqueue *vq,
                   struct scatterlist sg[],
                   unsigned int out_num,
                   unsigned int in_num,
                   void *data);
    void (*kick)(struct virtqueue *vq);
    void *(*get_buf)(struct virtqueue *vq,
                     unsigned int *len);
    void (*disable_cb)(struct virtqueue *vq);
    bool (*enable_cb)(struct virtqueue *vq);
};
```

VirtIO Transport layer - Vring

```
struct vring_desc
{
        __u64 addr;
        __u32 len;
        __u16 flags;
        __u16 next;
};
```

Vring : Descriptor chain

```
struct vring_avail
{
        __u16 flags;
        __u16 idx;
        __u16 ring[NUM];
};
```

Vring : available chain

NB : buffers should be saved in scatterlists before being recorded in virtqueues
We create as much scatterlists as we need to perform multiple requests.
This will generate the same number of descriptor chains.

Data exchange flow

- Data saved in a virtqueue
- The driver kicks the device through an event
- The device reads the virtqueues to get the buffers and process data.

NB : the driver first saves the data to the devicee in the virtqueue, and then put the buffers where the device should put the results back if any.

- The device process data and then triggers the guest.

Virtio transport wrapper

- In order to be able to communicate with the virtio device, there should be a channel.
- The three channel that exists nowadays are :
    - PCI channel : The standard PCI communication, to beneficiate from the whole PCI interface for devices.
    - MMIO channel : a memory mapped area dedicated to that device in the guest memory.
    - Channel I/O available on certain devices only

Go through : virtio example

Example of Virtio driver :
https://github.com/ybettan/QemuDeviceDrivers/blob/master/virtio/virtio_example_driver.c
Example of virtio device :
https://github.com/ybettan/qemu/blob/virtio/hw/virtio/virtio-example.c

virtio input specifications (some)

Usage

This is used to virtualize an interface that might serve to put data into the virtual machines. An example is the keyboard.

Device ID: 18

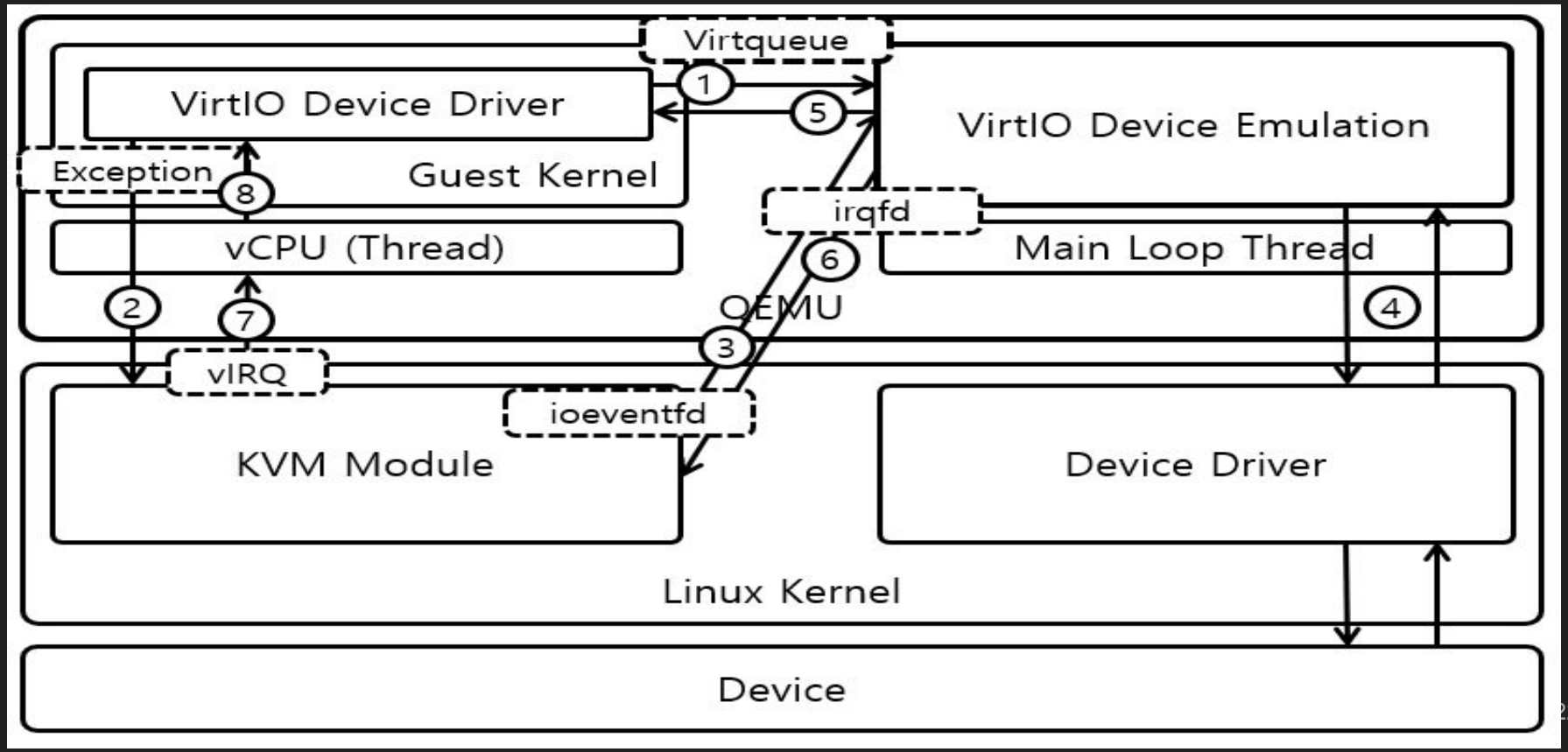
Virtqueues

eventq : virtqueue used to receive input
statusq : notify for the status

Feature bits

None

Summary (using QEMU)



VHOST

Vhost

- Vhost: kernel-space implementation of virtio
- Kernel thread moving data from/to the VQ
- Does not avoid VM exits, but avoids KVM exits
- Can avoid a lot of kernel-space/user-space switches
- Can improve virtio throguhput (and reduce latency) by moving functionalities to the kernel
- There remains only the control plane in the userspace part of the hypervisor

TP : CAESAR CIPHERING DEVICE

TP : CAESAR CIPHERING DEVICE

Link : https://github.com/brisco007/tp-virtio-ens

REFERENCES

- https://docs.oasis-open.org/virtio/virtio/v1.1/csprd01/virtio-v1.1-csprd01.html OASIS Virtio documentation
- https://developer.ibm.com/articles/l-virtio/ Virtio implémentation in Linux
- The inspiration for these slides : https://www.cs.cmu.edu/~412/lectures/Virtio_2015-10-14.pdf