Technology Consulting Company
Research, Development &
Global Standard

Input Device Interrupt Latency of KVM on ARM using Passthrough

Koji Teranishi
寺西　光司
IGEL Co., Ltd.

Who are you?

- Software engineer
- Recent work
  - Survey around using kvm on arm
  - Developping application with Qt, web languages

Overview

- Introduction (about KVM ,device virtualization,...)
- Pareparation of environment
- Implementation for measurement
- Result
- Case study

Introduction

Introduction

- KVM on ARM
  - KVM(Kernel-based Virtual Machine) is a kernel module that enables the kernel to run as a hypervisor
  - KVM supports processor of intel, AMD, also ARM
  - KVM is used with machine virtualizer like QEMU

Introduction

- Three types of Device virtualization
  - Full virtualization
    - Hypervisor provides a virtual device completely emulates a real hardware device
    - Low performance by the emulation
  - Paravirtualization
    - Hypervisor provides a virtual device which is optimised for virtual environment
    - The guest needs a device driver designed for the device
    - Improve performance compare to full virtualization
  - Passthrough
    - Hypervisor exposes a real device to the guest exclusively
    - Guest can directly use the device as if on the native environment
    - Get high performance near the native environment

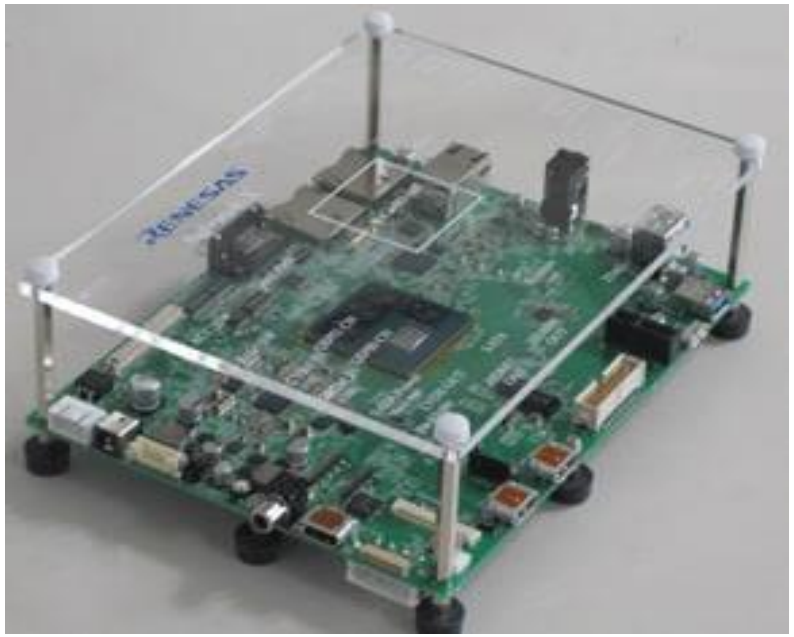
Introduction

- Some latency will occur on using passthrough, but how much is it ?
- Want to measure the latency time of USB HID passthrough
- The processing of virtualization can give some latency to the usb host controller
- We try to measure the latency on the interrupt handler

Preparation of environment

Preparation of environment

- Renesas R-Car M3 (R8J77960(SiP))
  - Arm Cortex A-57 dual,53 quad, -R7 Dual Lock-Step
  - IMG PowerVR Series6XT GX6250
  - Memory controller for LPDDR4-SDRAM 32bit bus 2ch
  - 3 channels Display Output
  - 8 channels Video Input
  - USB3.0 and USB2.0(OTG) interfaces
  - PCI Express Interfaces

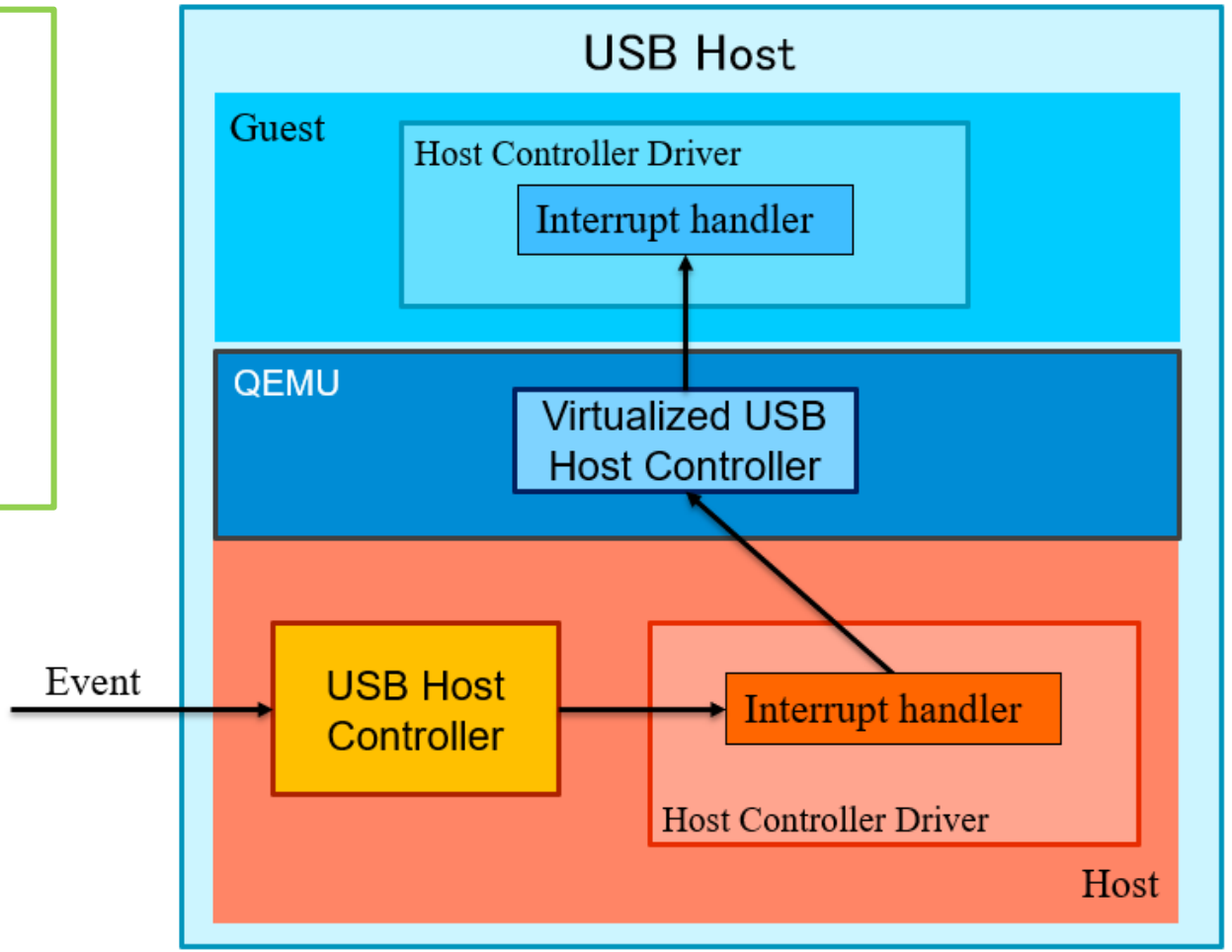


Preparation of environment

We measure the interval between host controller driver's interrupt handler of Host and Guest

■ The order of event step

1. USB Host Controller
2. Interrupt handler on Host
3. Virtualized USB Host Contorller
4. Interrupt handler on Guest



Preparation of environment

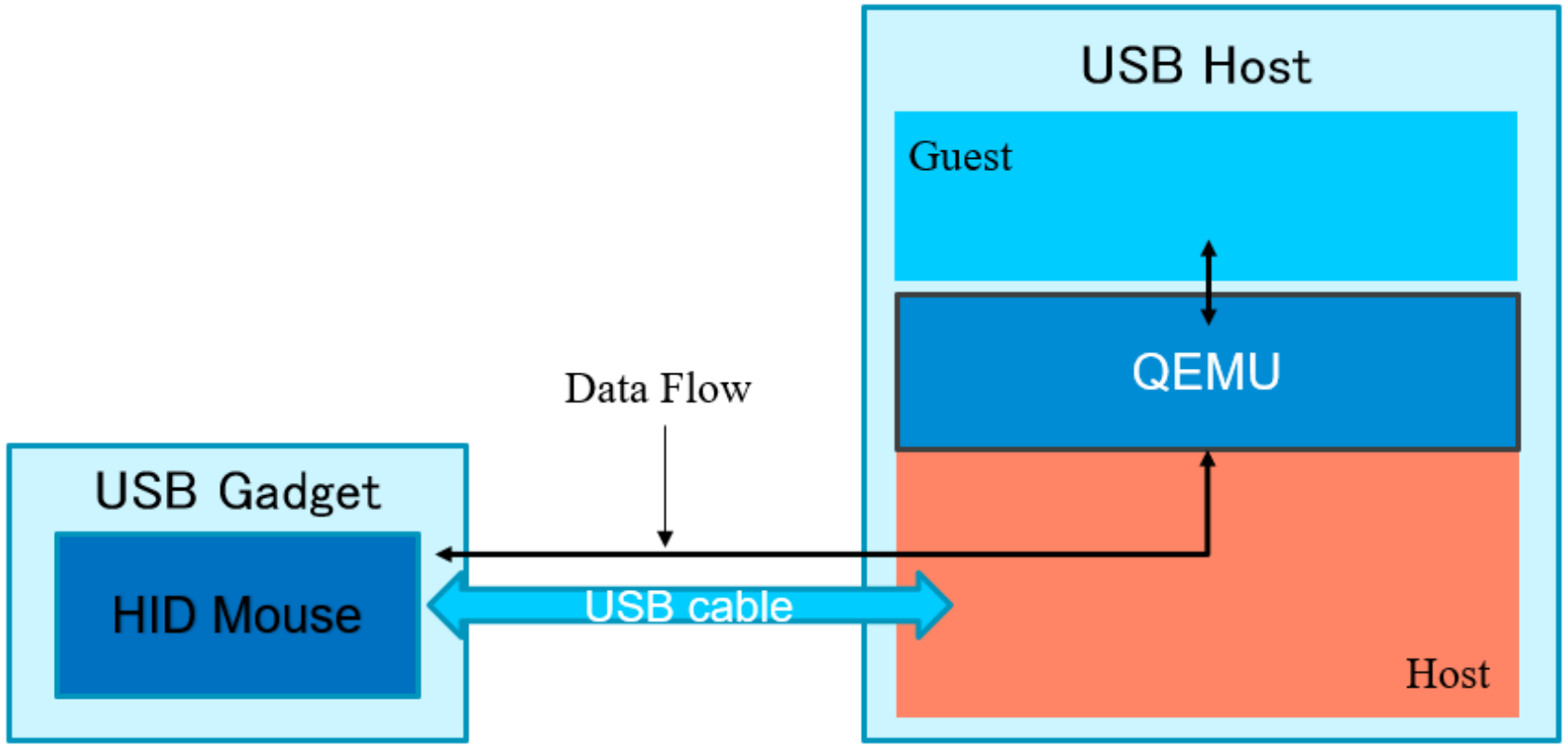
- What type of HID device should we use for sending event ?
  - Want to measure the latency closely
  - Process like sending many events at a regular intervals will be needed
  - Want to control of the sending events by programming…

Normal HID device (like mouse,keyboard …) is not enough

Using the board as HID gadget is an answear (the board has USB OTG)

Preparation of environment

- Using two board (as HID Gadget and the Host)
- The type of the gadget is mouse(enough for sending a data)



Preparation of environment

- Some clock is needed to measure the time
- The system clock which runs on host is not common to guest's
- We need a clock which is independent from them

The board we use have a built-in hardware timer module (called TMU)

Preparation of environment

- Setup USB gadget
  - Need to enable HID Gadget on kernel config of USB gadget kernel or enable below parameters

  - CONFIG_USB_GADGET=y
  - CONFIG_USB_F_HID=y
  - CONFIG_USB_G_HID=y
  - USB_LIBCOMPOSITE=y

Preparation of environment

- Setup USB gadget
  - Write a script to setup a simple HID mouse gadget

  https://github.com/qlyoung/keyboard-gadget is useful

```
#!/bin/bash
PROTOCOL=2
SUBCLASS=1
REPORT_LENGTH=8
UDC=e6590000.usb
…
```

Replace UDC driver name at /sys/class/udc/

  - Run the scrupt

  If it succeeds , we can find the device file /dev/hidg0 !

Implementation for measurement

Implementation for measurement

- Kernel
  - The USB host controller driver need to handle TMU when a interruption happens
  - On the Host, it resets and starts the timer
  - On the Guest, it gets the time and output the result by printk

Implementation for measurement

- QEMU(virtual machine emulater)
  - Memory mapping is needed to show host's TMU register to guest
  - Add a virtual device which do the mapping to qemu code
  - The file hw/misc/exynos4210_pmu.c was helpful

The example of memory mapping part

We'll set TMU's

```
int fd = open("/dev/mem", O_RDWR|O_SYNC);
mapped_addr = mmap( …, fd, the offset we'll access);
memory_region_init_ram_ptr(region, … , device_name, region_size, mapped_addr);
memory_region_add_subregion(get_system_memory(), … , region);
```

Implementation for measurement

- What we need for script of starting QEMU
  - USB host controller
  - Host device to be passed
  - Virtual device we made

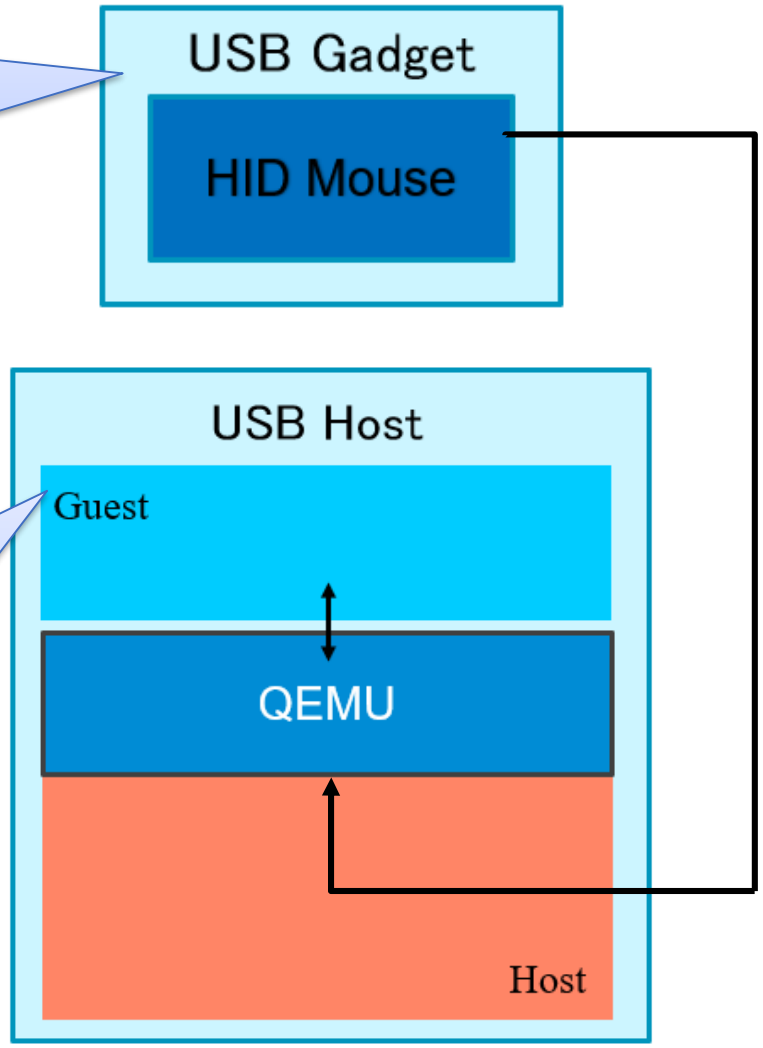
```
qemu-system-aarch64 ¥
...
-device nec-usb-xhci,id=xhci ¥
-device usbhost,hostbus=1,hostport=1.3 ¥
-device device_name
```

The bus and port information can be found by “lsusb -t” command

Implementation for measurement

■Program sending event
- Open the gadget device file(/dev/hidg0)
- Write a event data to the fd

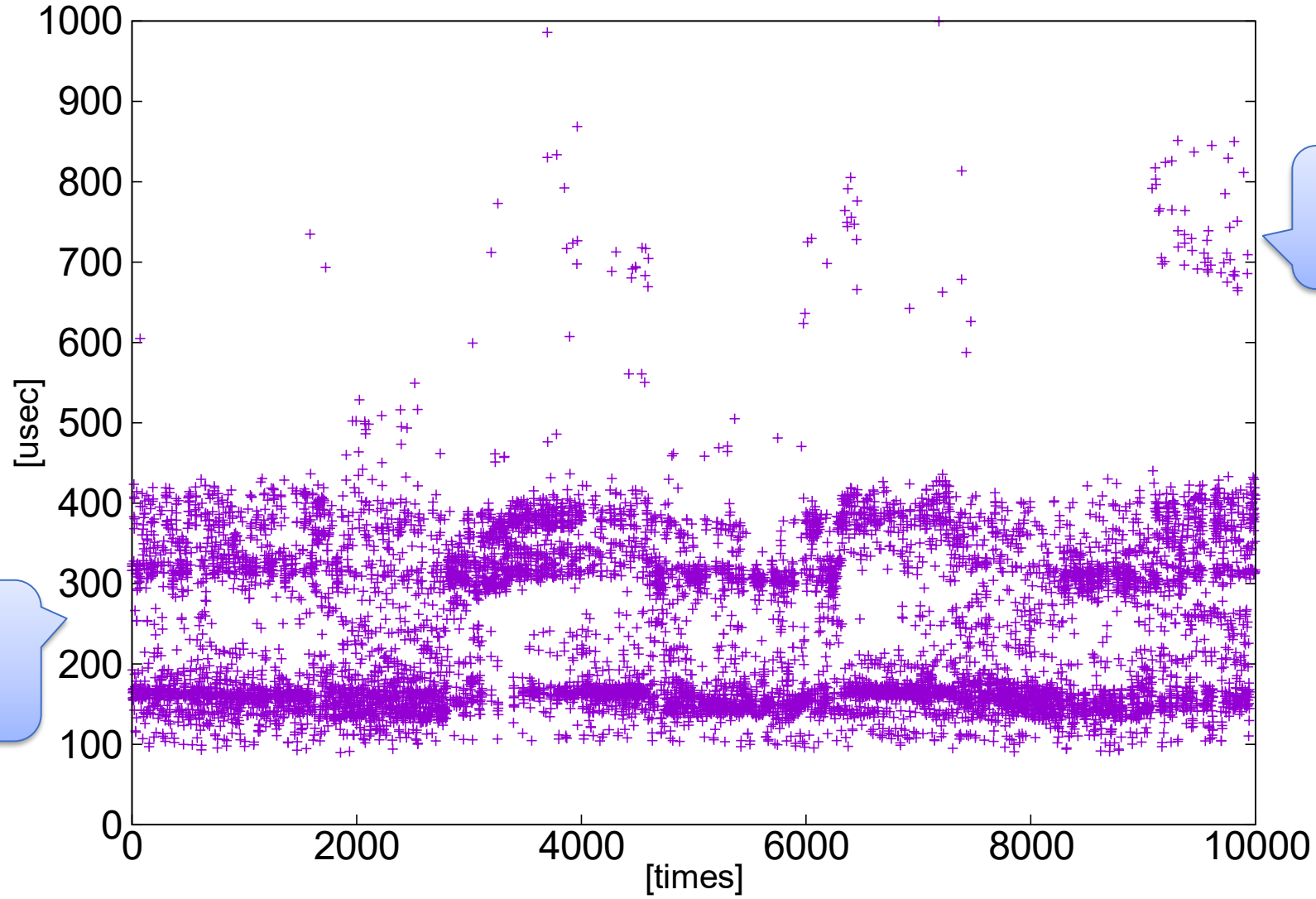
If the program start …
- The results can be shown by “dmesg”
- “dmesg -w” will follow the messages continually
- Redirect the result to a file



Result

Result

- The result of sending 10000 times with 10ms intervals

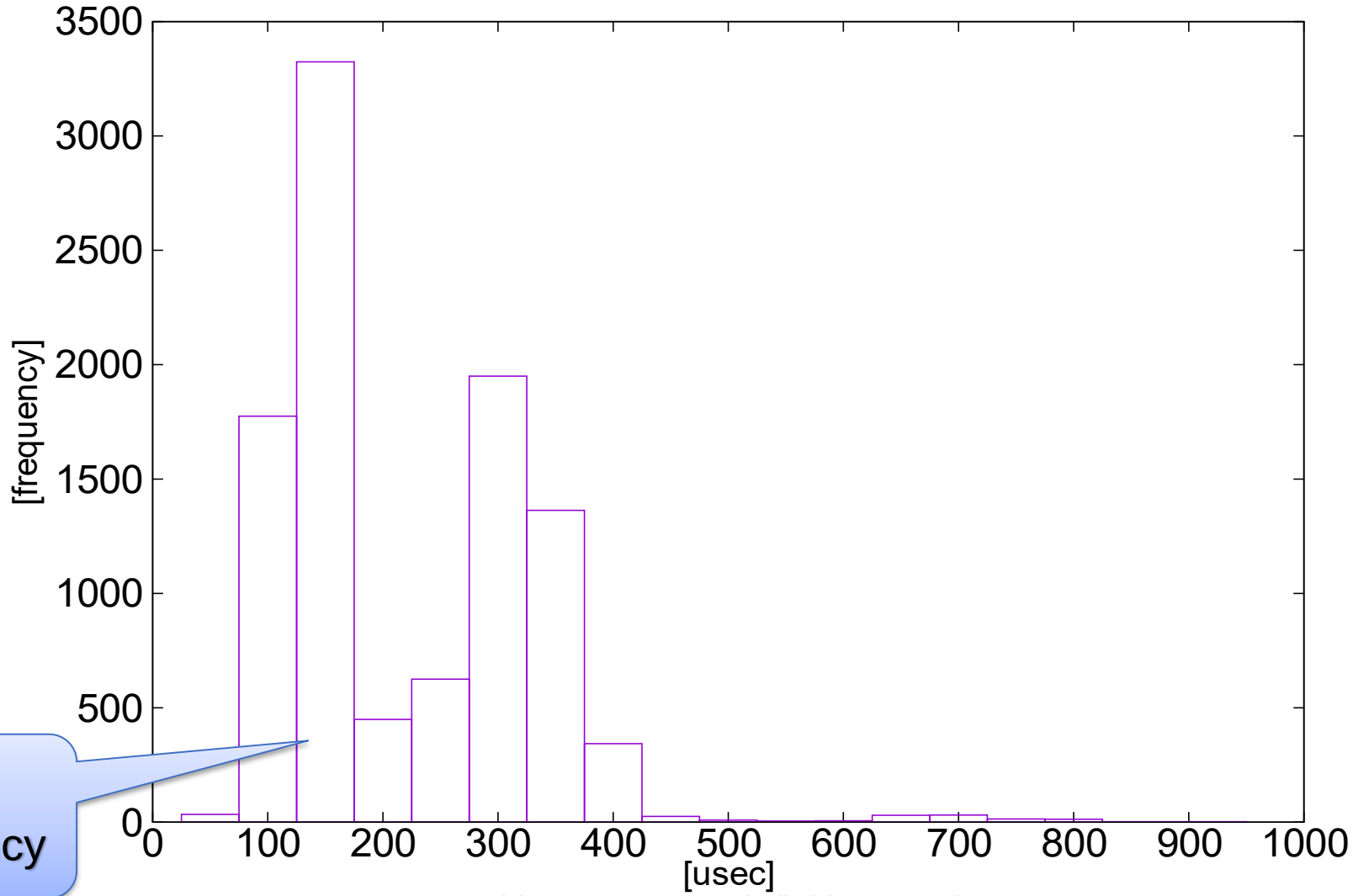


Very small amont is high latency

Almost around 100 ~400

Result

- The histogram of the frequency with 50 usec intervals



150~200 is highest frequency

Case study

Case study

- Examined the latency of one direction(host -> guest)
- How about it containing reverse(guest -> host)?
- How to measure
  - Try to measure the latency on using keyboard
  - By pressing CapsLock, we get feedback to turn on/off LED
  - Measure the roundtrip time
  - Compare the result of host(no virtualization) to guest

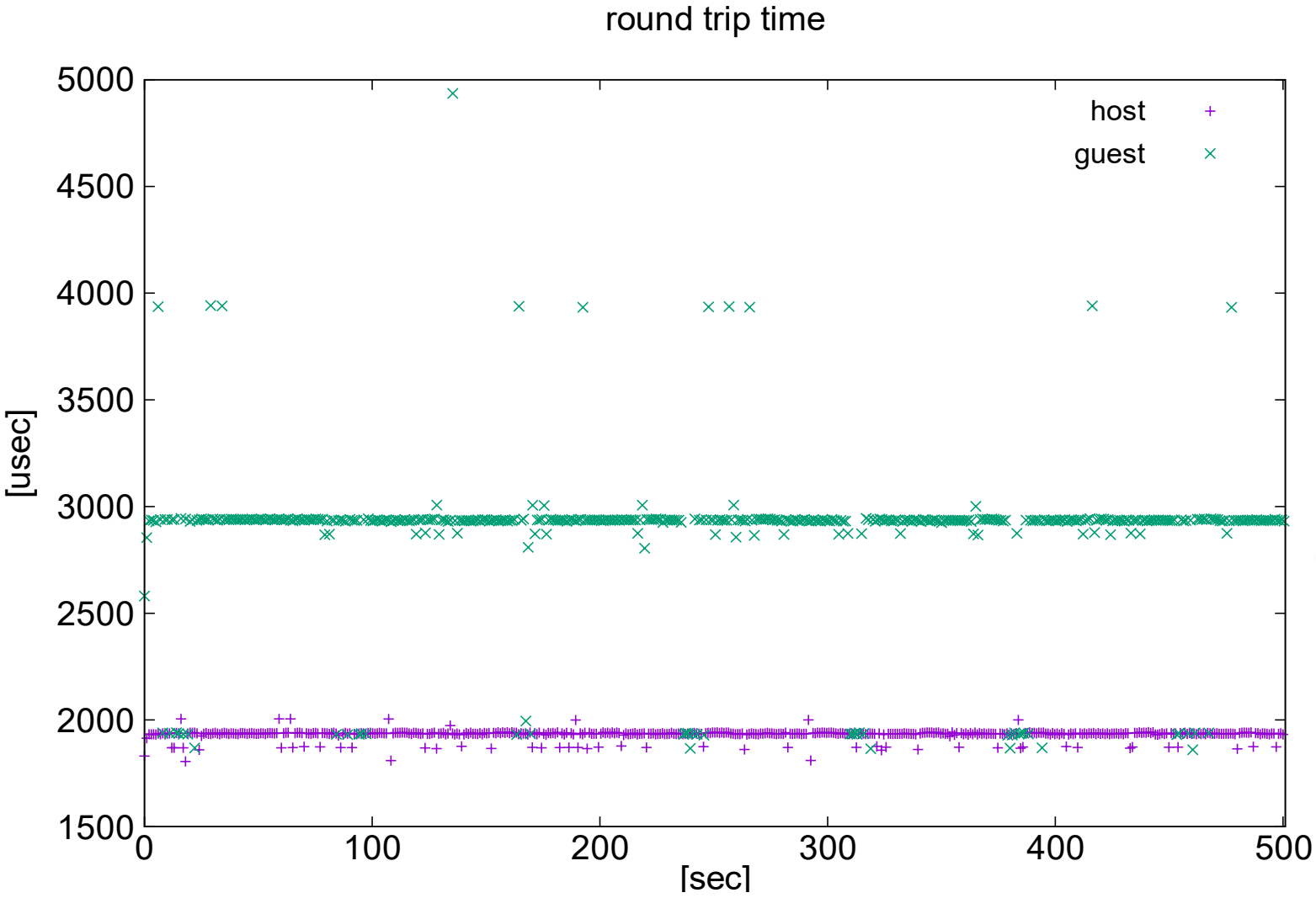
Case study

- Preparation
  - Same environment to previous one but use HID Keyboard as gadget

- Detail
  - Send a key data at a regular intervals
  - Wait until getting feedback
  - Measure the roundtrip time by using clock_gettime function

Case study

- Result – Sending 500 times every 1 second



About 1ms later than host

Conclusion

Conclusion

- The latency between interrupt handlers was almost 100 ~200 usec
- It was not significant latency in the case of HID device
- The result of roundtrip seems to be affected by USB host polling to device every 1ms on HID

Title