HA-IOV

Applying Hardware-assisted Techniques to I/O Virtualization Framework

Yifei Jiang <jiangyifei@huawei.com>

Bo wan <wanbo13@huawei.com>

HUAWEI

Contents

Background

- High-performance computing in data centers
  - I/O virtualization is one of the most crucial components to optimize physical resource utilization as well as I/O performance of VMs

Background

- High-performance computing in data centers
  - I/O virtualization is one of the most crucial components to optimize physical resource utilization as well as I/O performance of VMs
- Existing I/O virtualization mechanisms
  - Hardware-assisted Techniques,
    - ✓ Such as, SR-IOV and IOMMU
    - ✓ Allowing VMs to direct pass-through access physical I/O devices.

    But **complicates live migration**.

Background

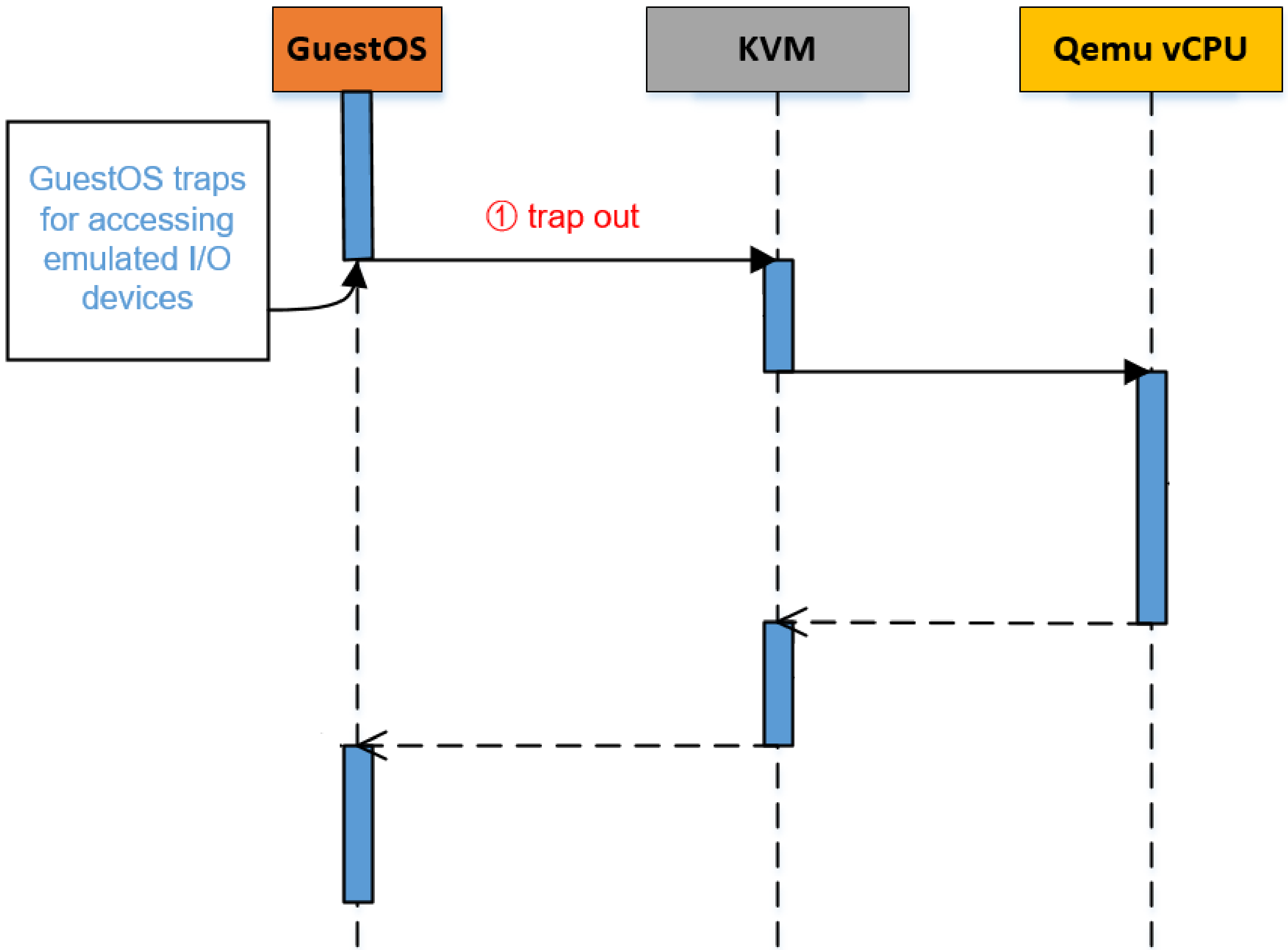
- High-performance computing in data centers
  - I/O virtualization is one of the most crucial components to optimize physical resource utilization as well as I/O performance of VMs
- Existing I/O virtualization mechanisms
  - Hardware-assisted Techniques,
    - ✓ Such as, SR-IOV and IOMMU
    - ✓ Allowing VMs to direct pass-through access physical I/O devices.

    But **complicates live migration**.
  - Software Techniques
    - ✓ Full emulated paradigm, such as UART in QEMU/KVMTOOL;
    - ✓ Paravirtual paradigm, such as VirtIO-blk, VirtIO-net, and vHost;
    - ✓ Polling mode, such as DPDK and SPDK;

    However, full emulated and paravirtual paradigm **suffer performance loss** due to costly context switches between Guest and Host, while polling mode **lowers the CPU utilization.**

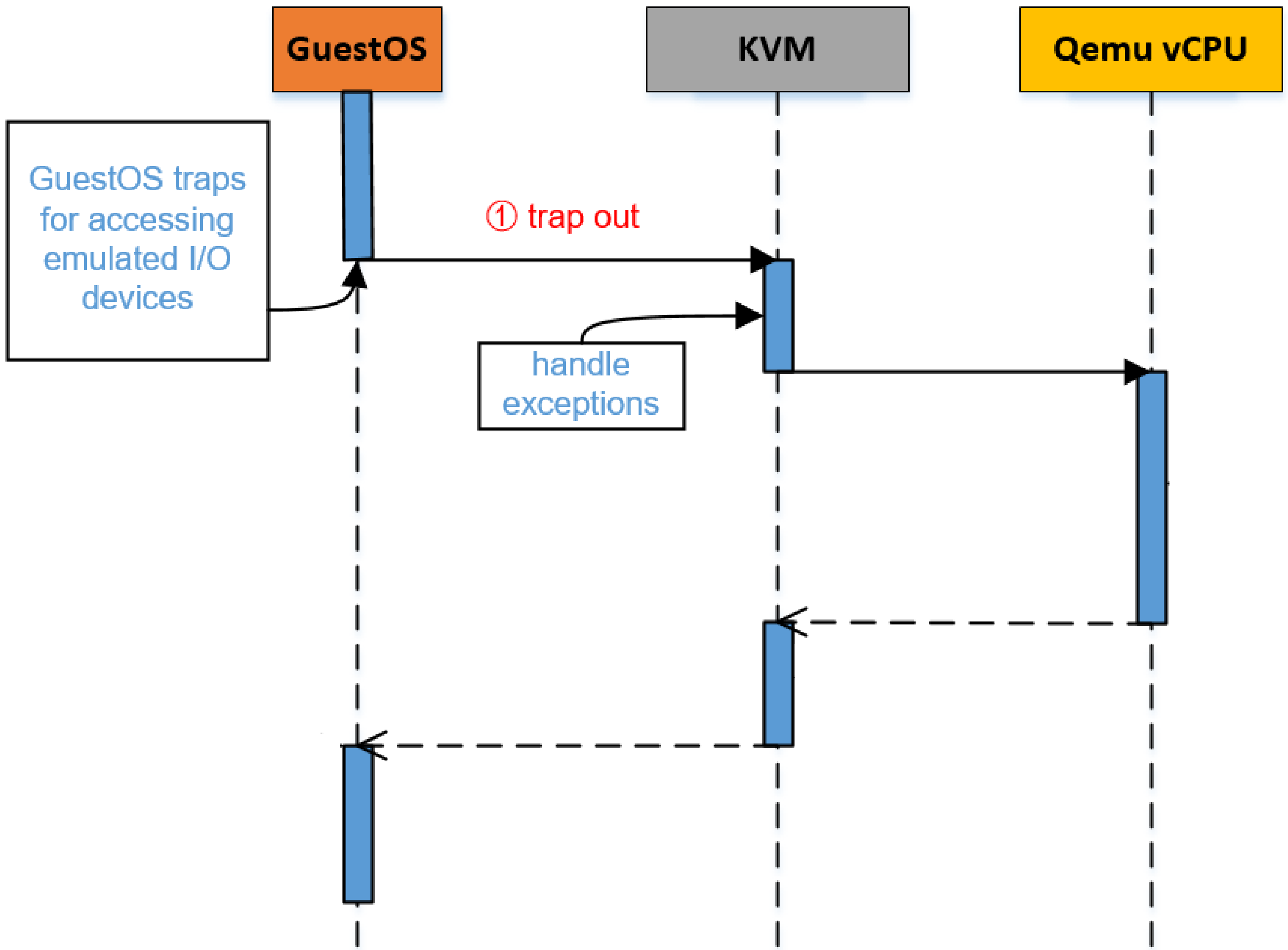
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



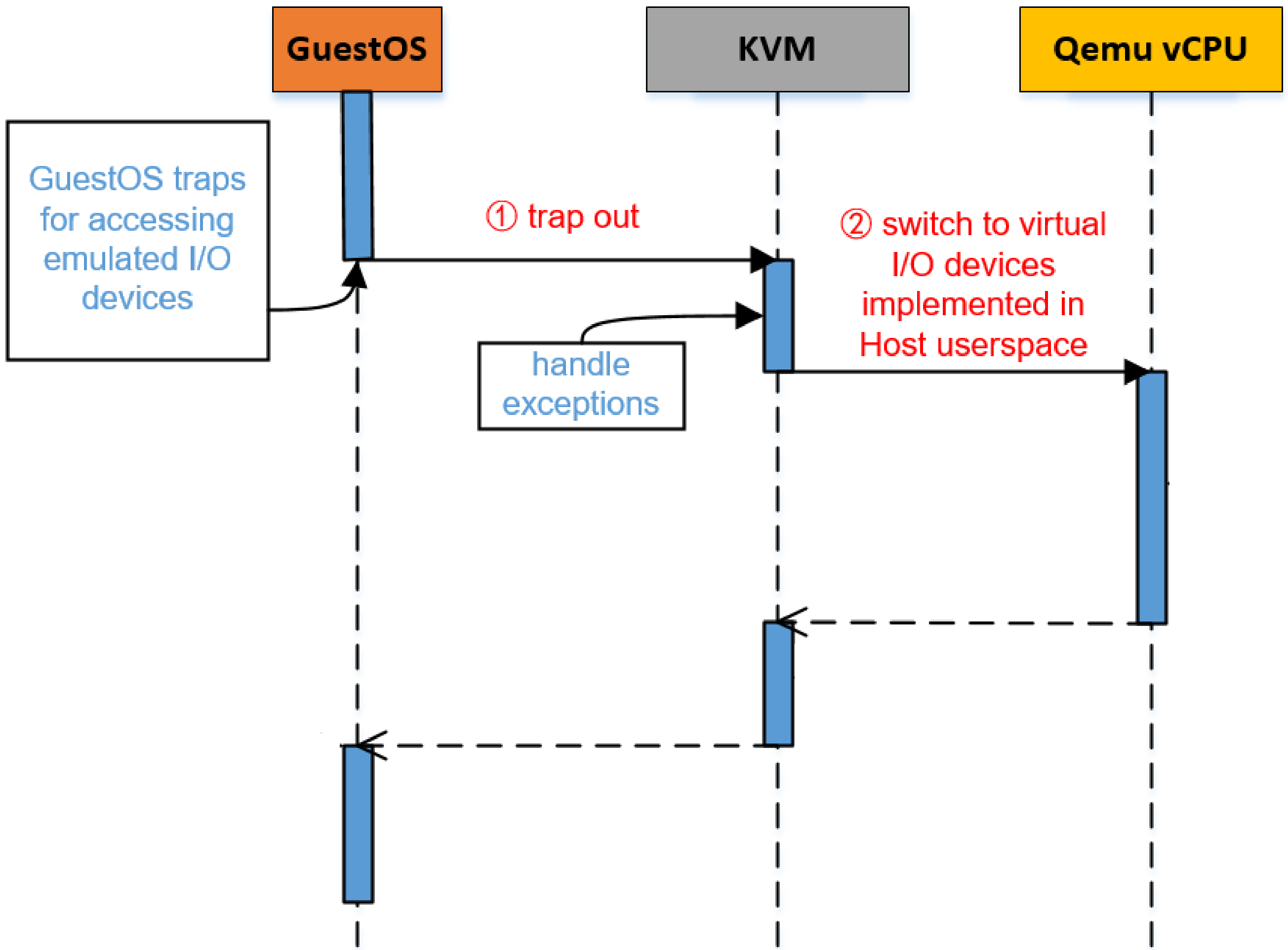
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



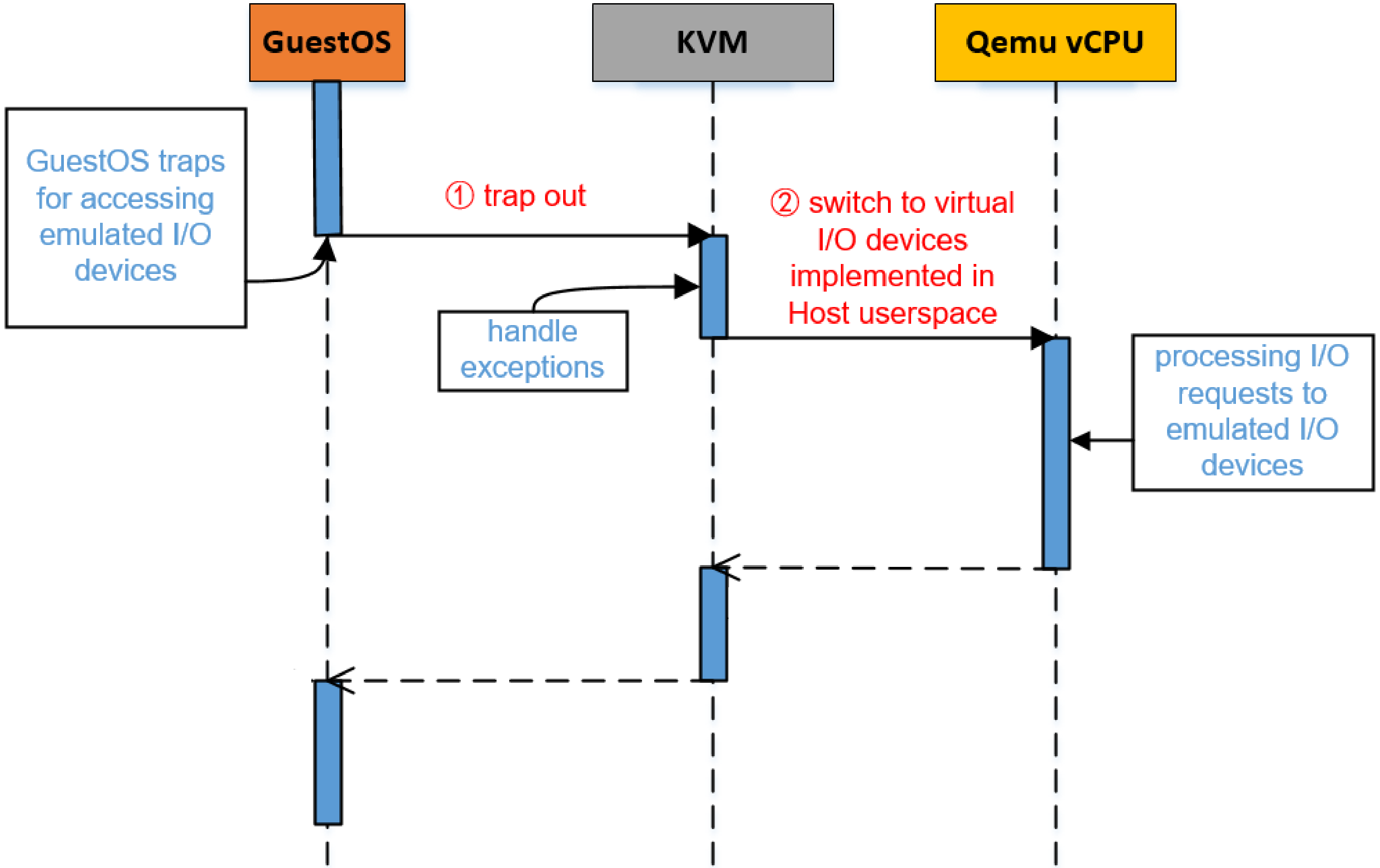
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



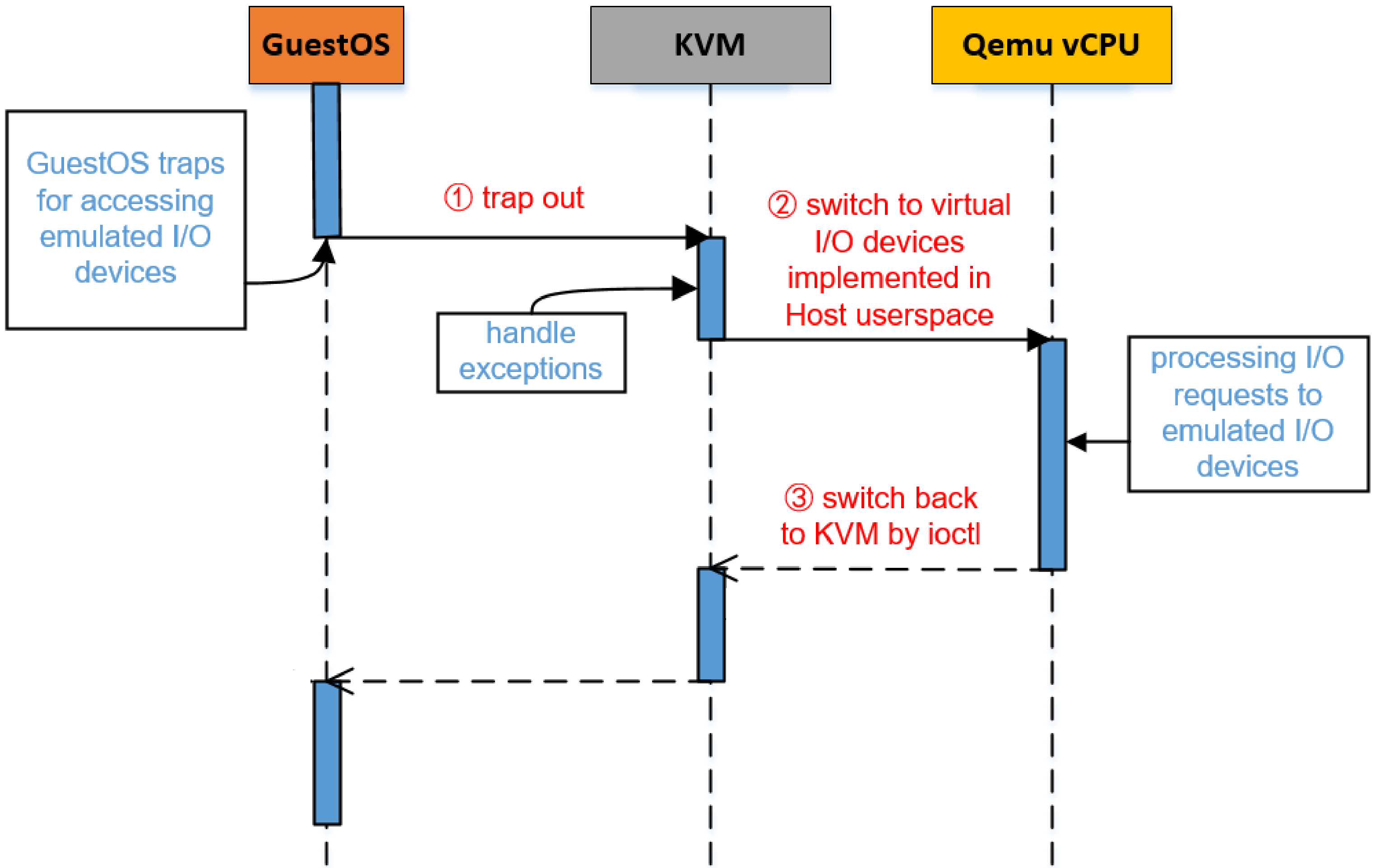
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



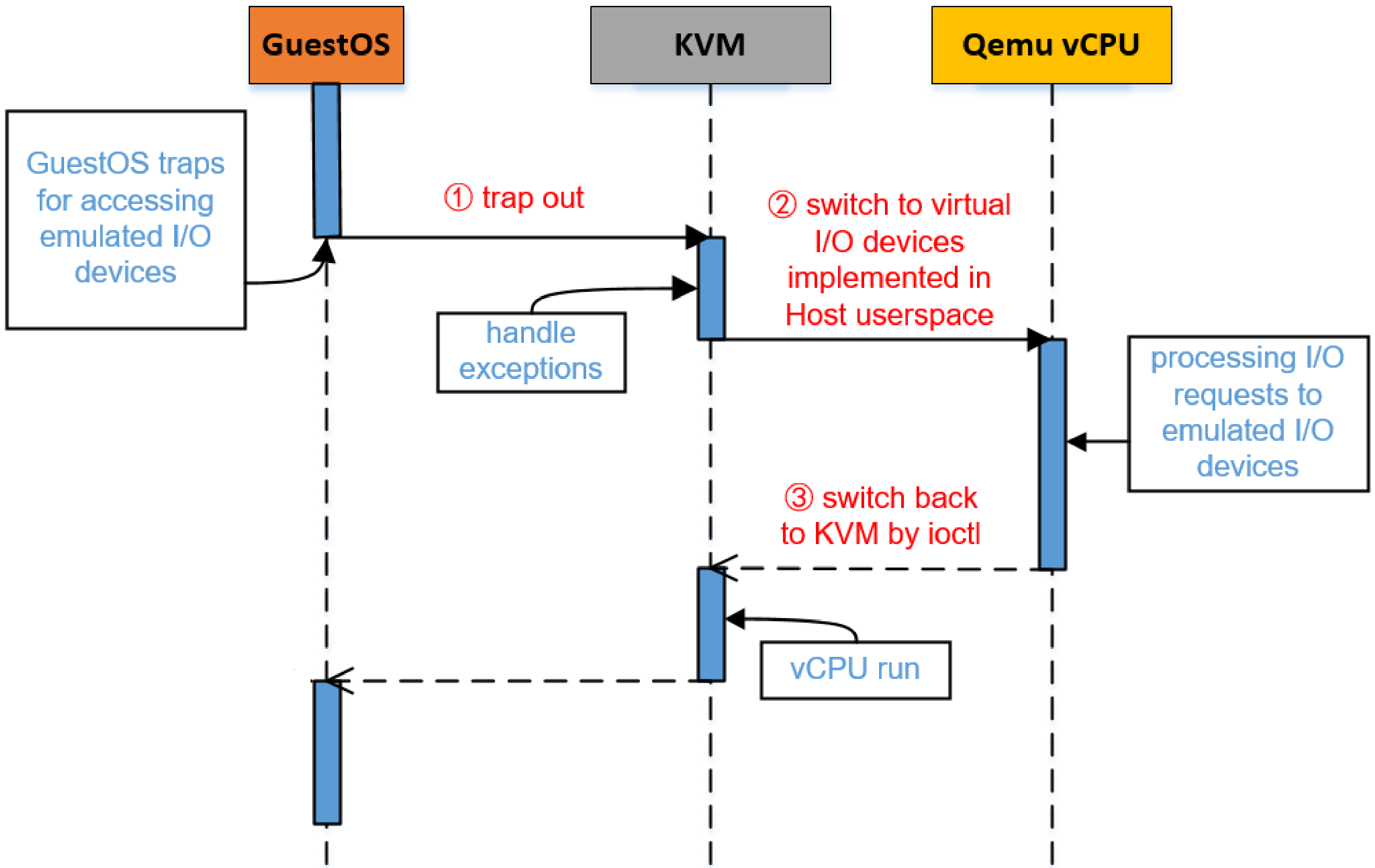
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



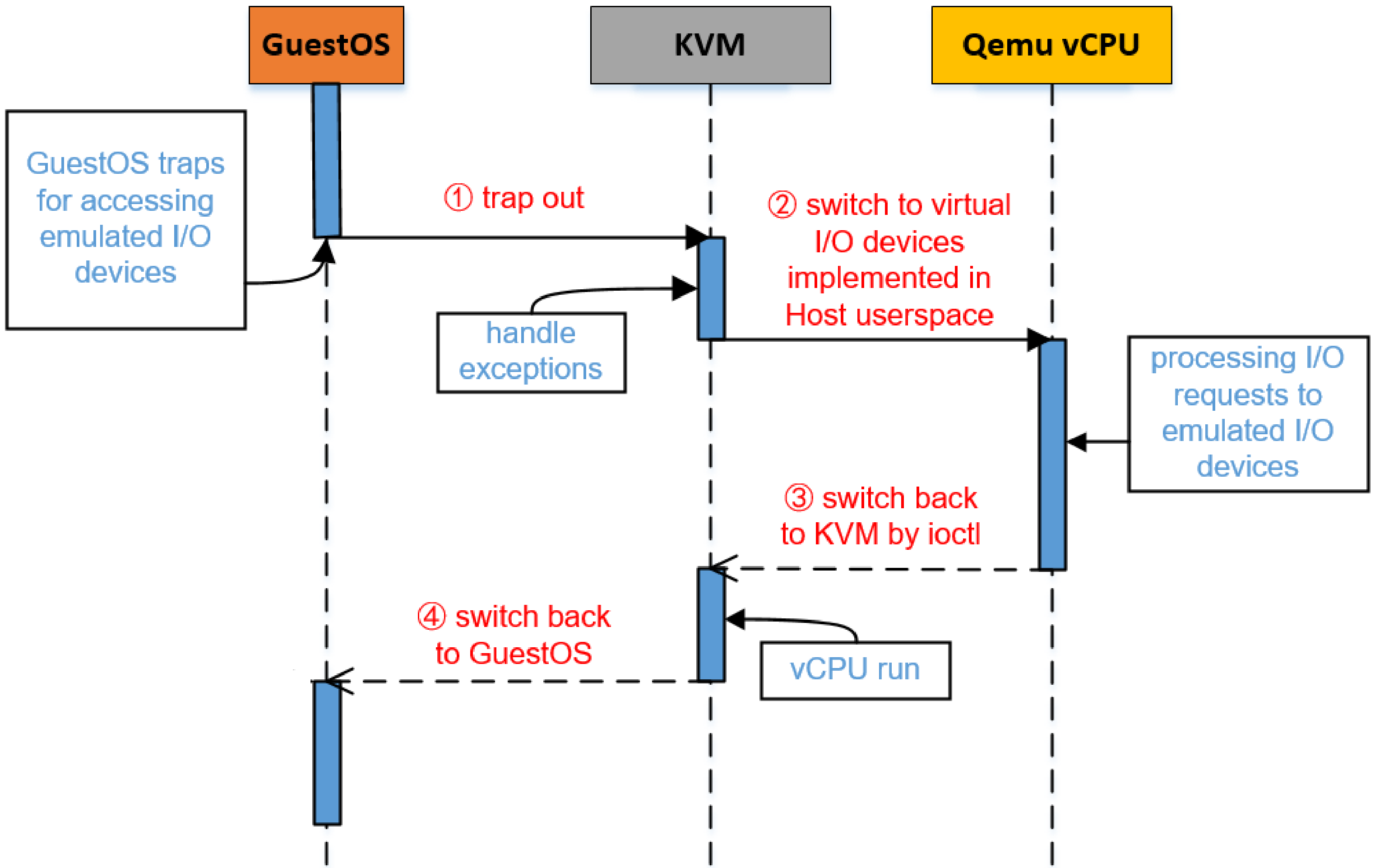
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



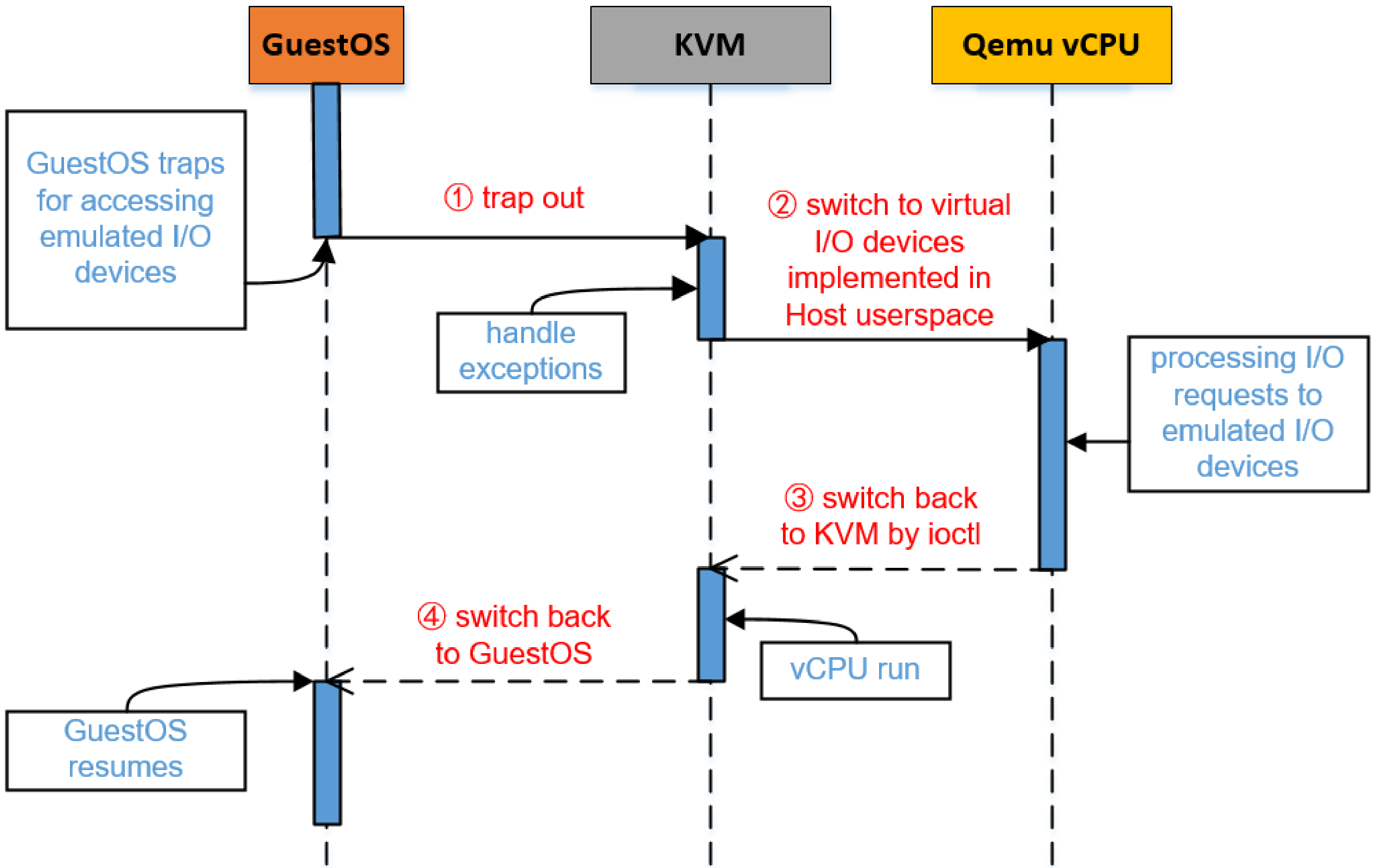
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



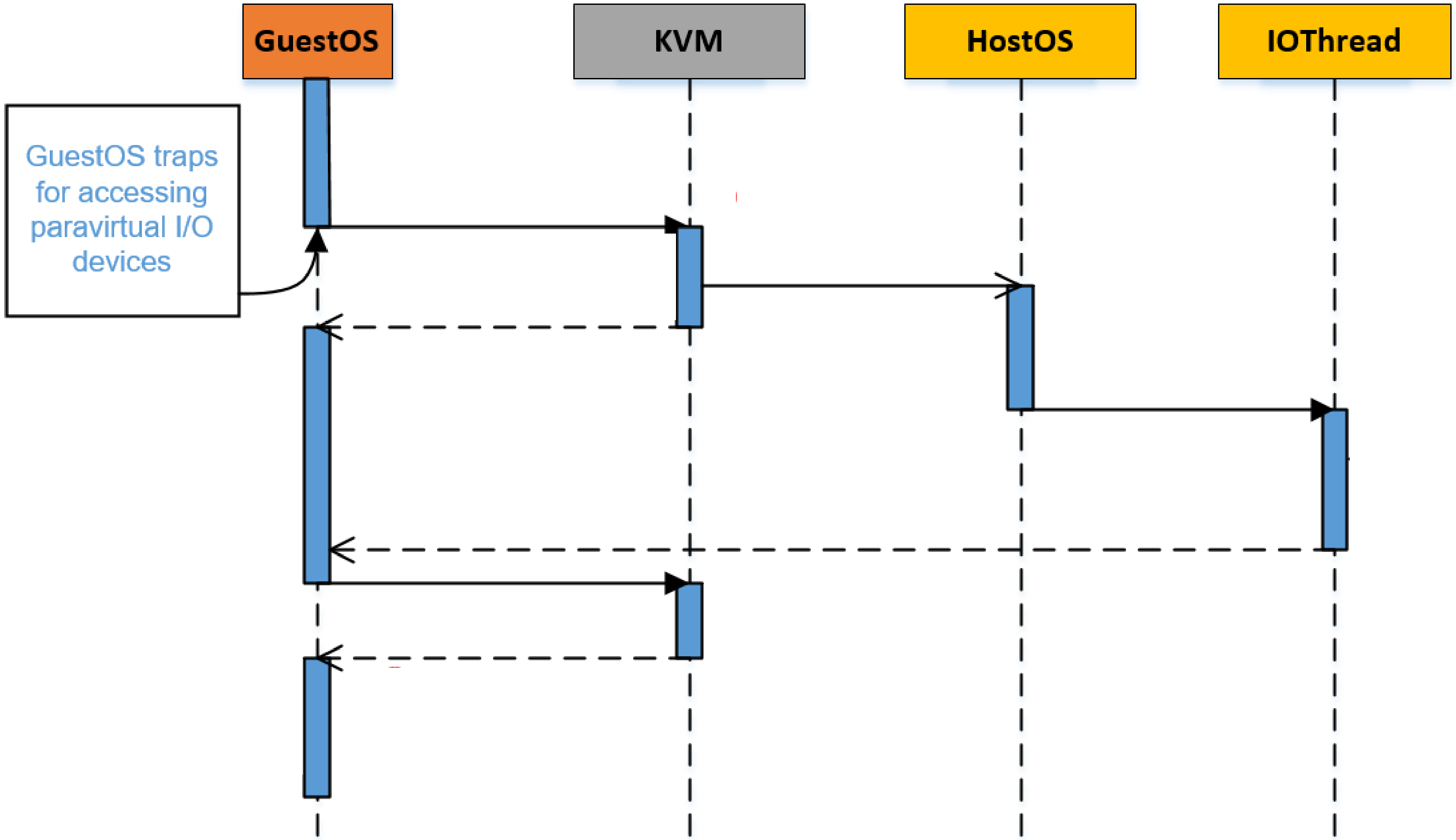
Motivation – Emulated Virtual I/O Devices

- Trap and Emulate in Qemu



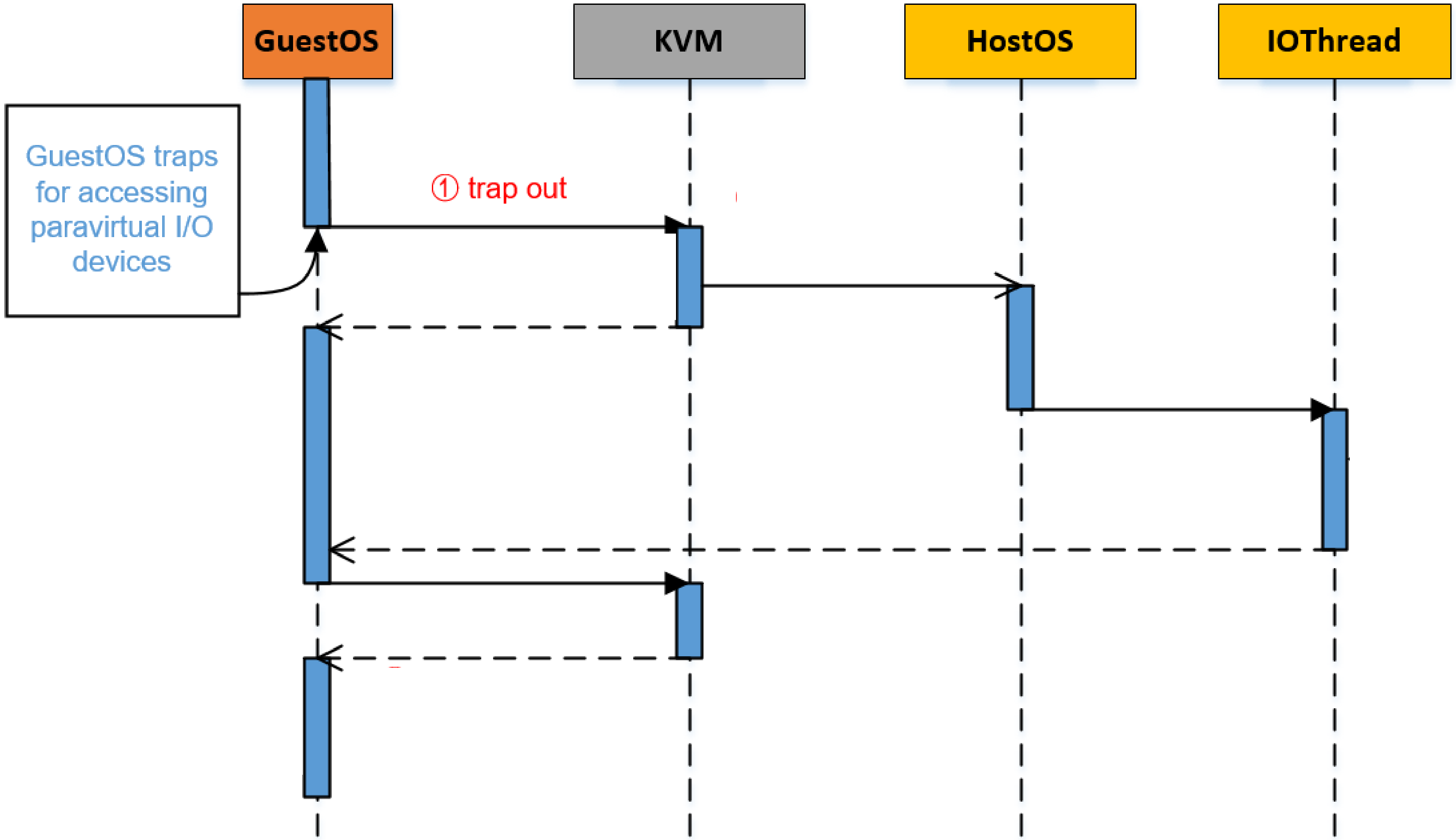
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



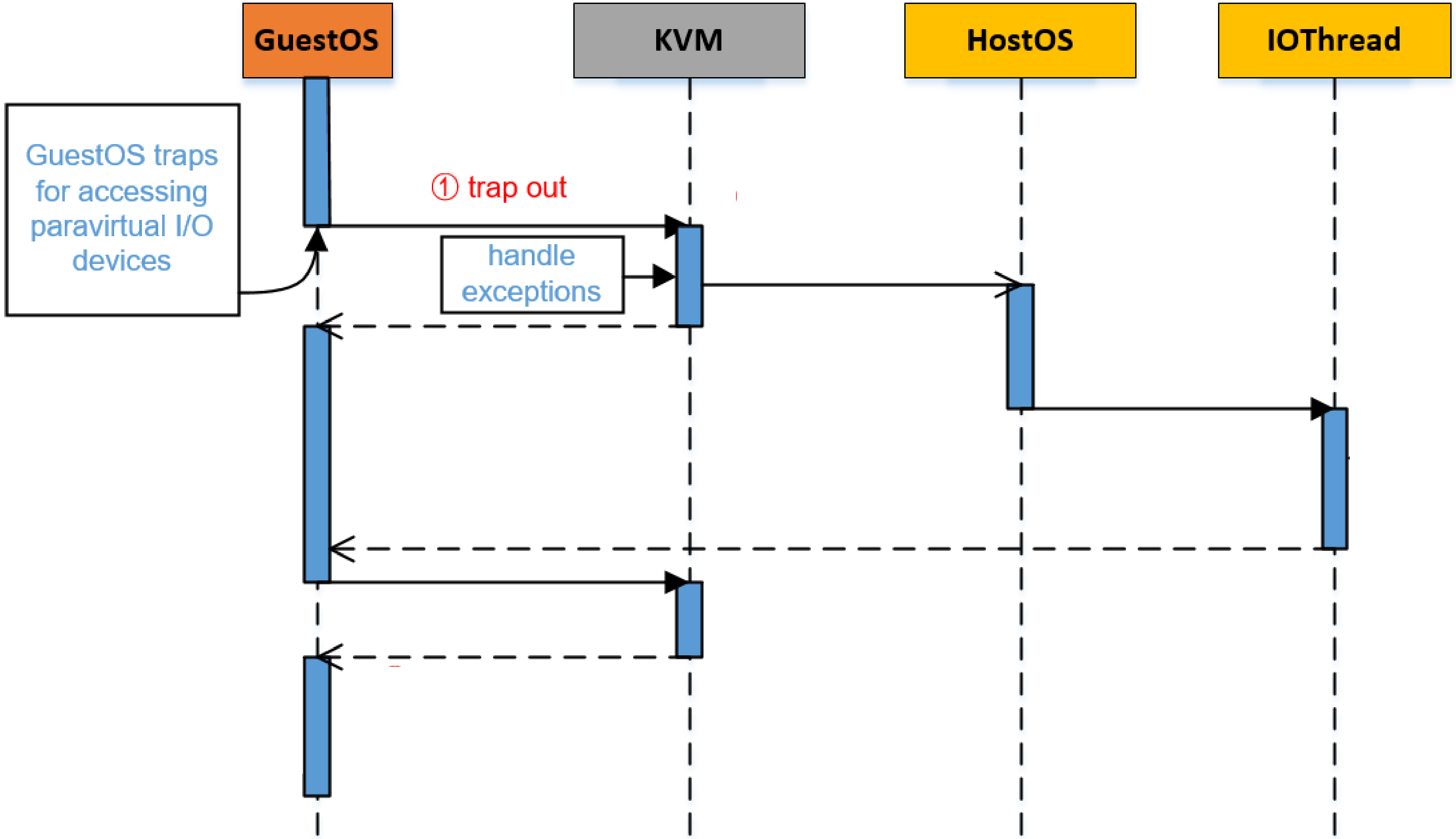
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



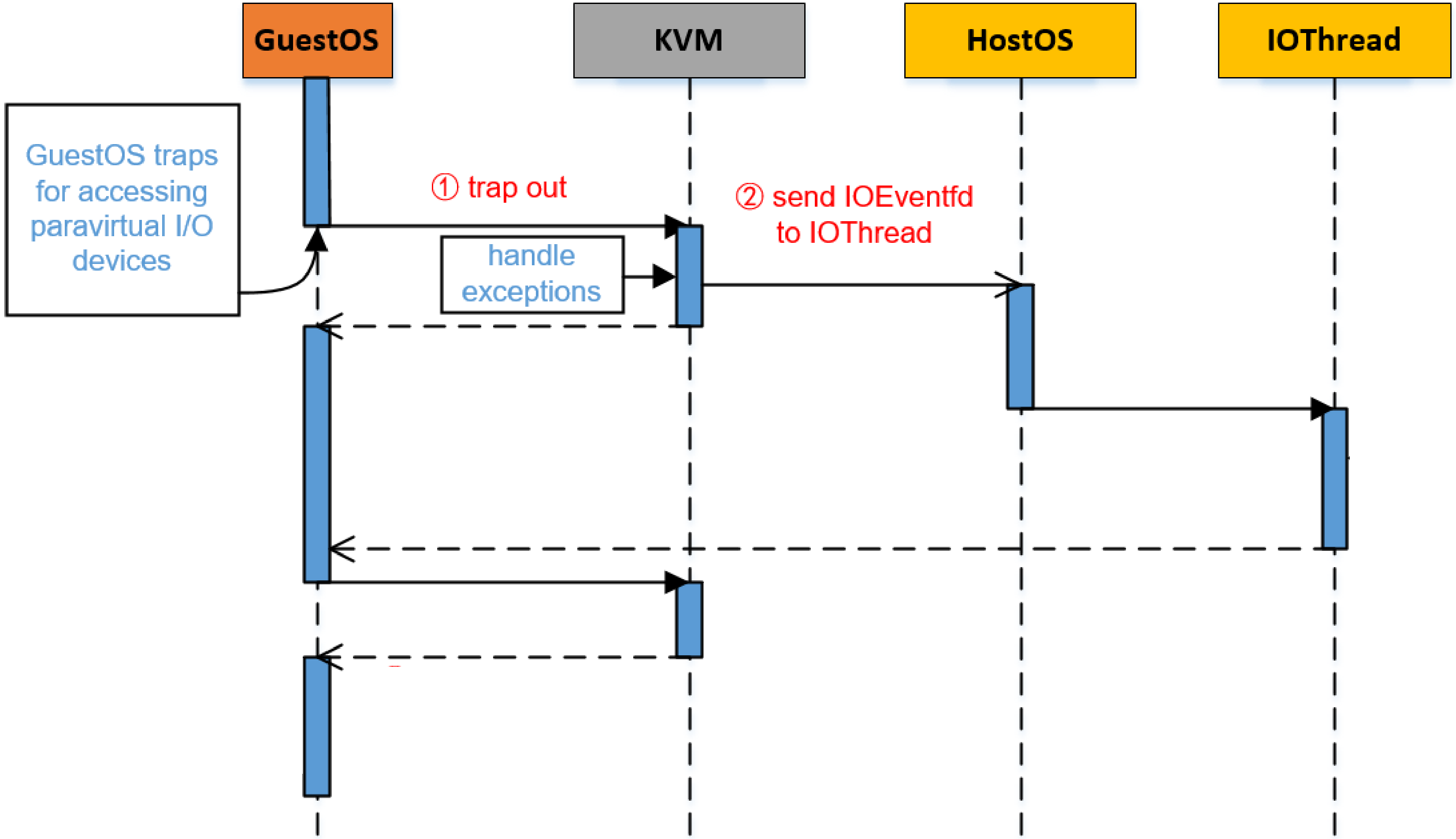
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



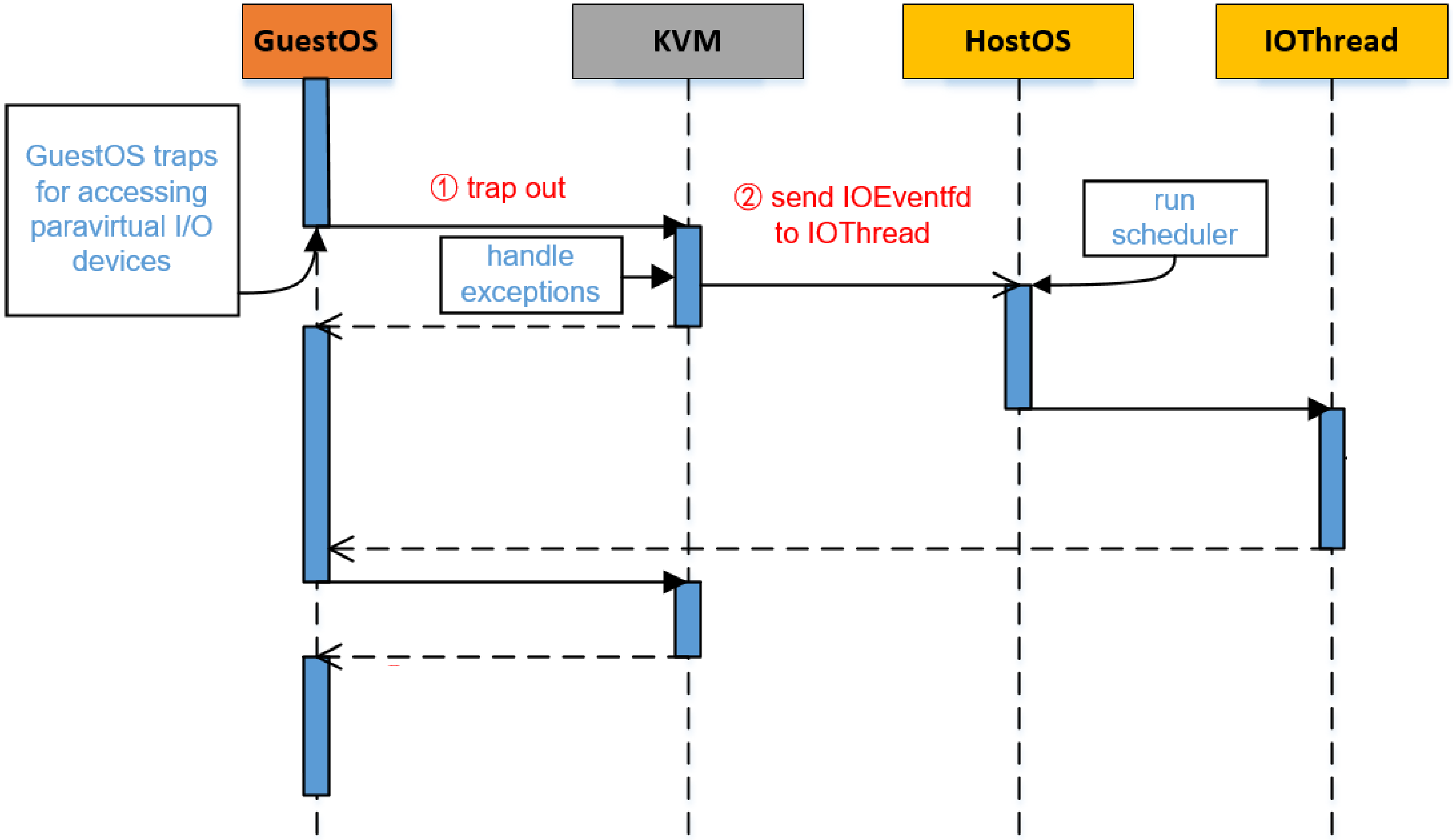
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



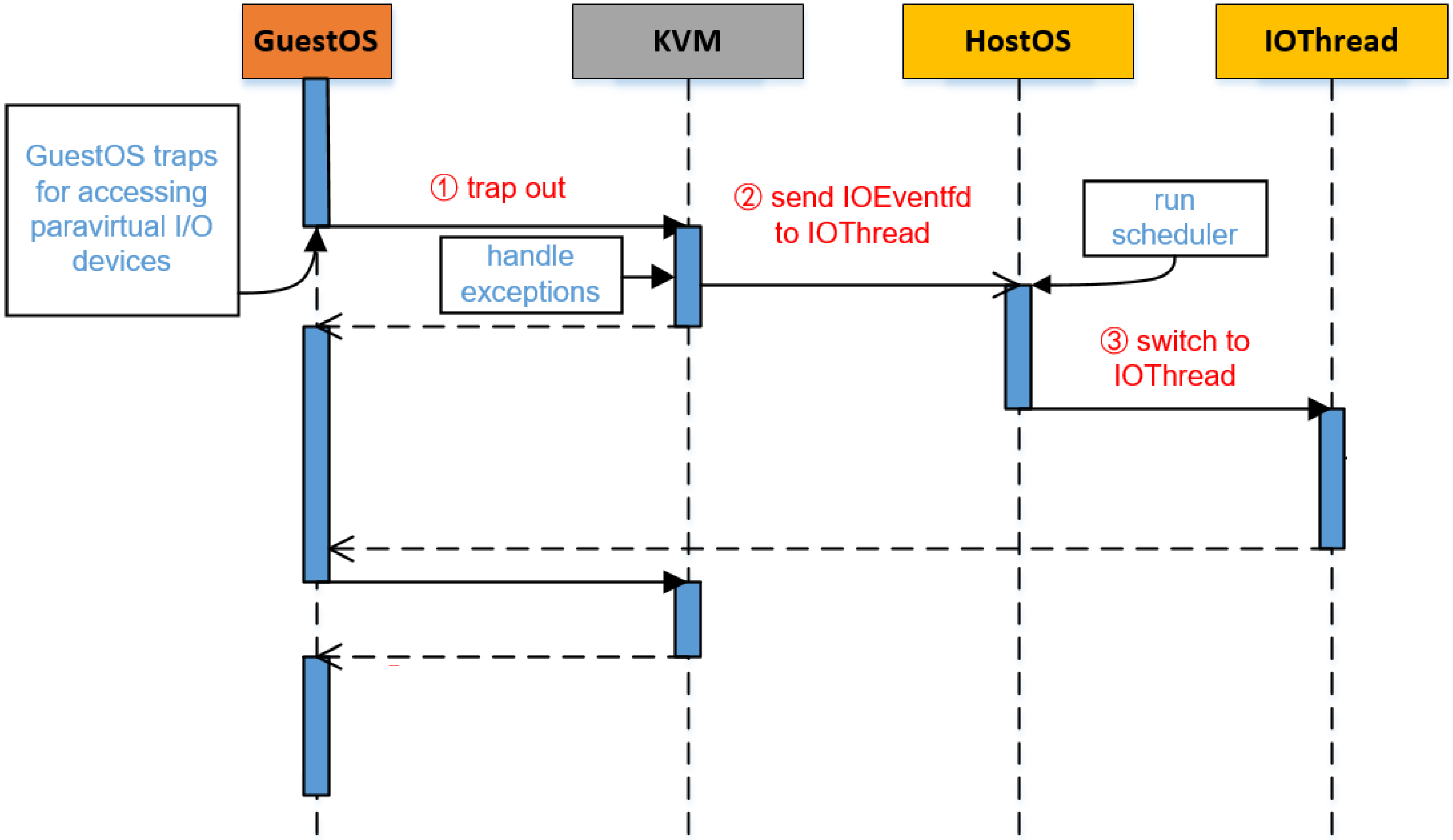
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



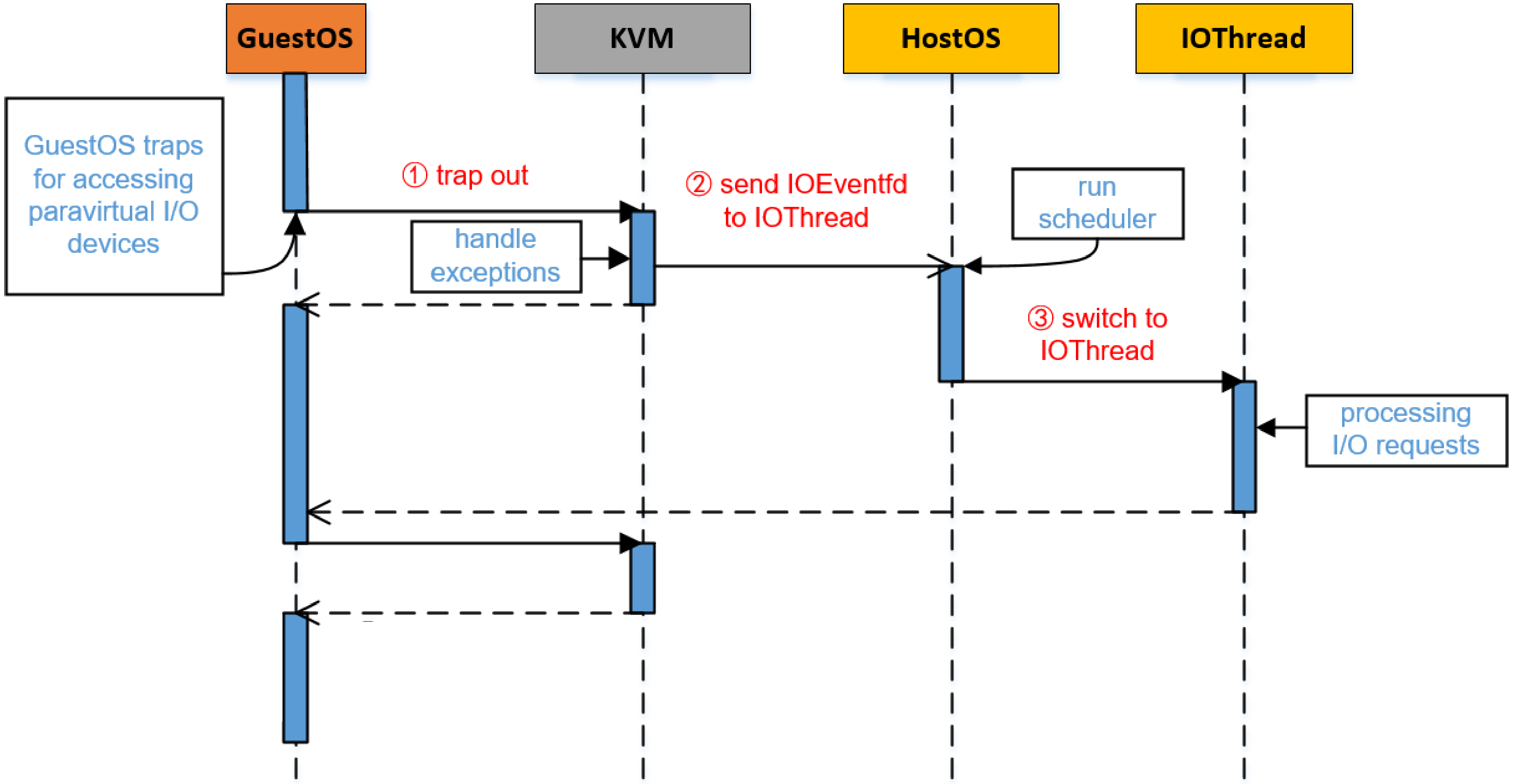
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



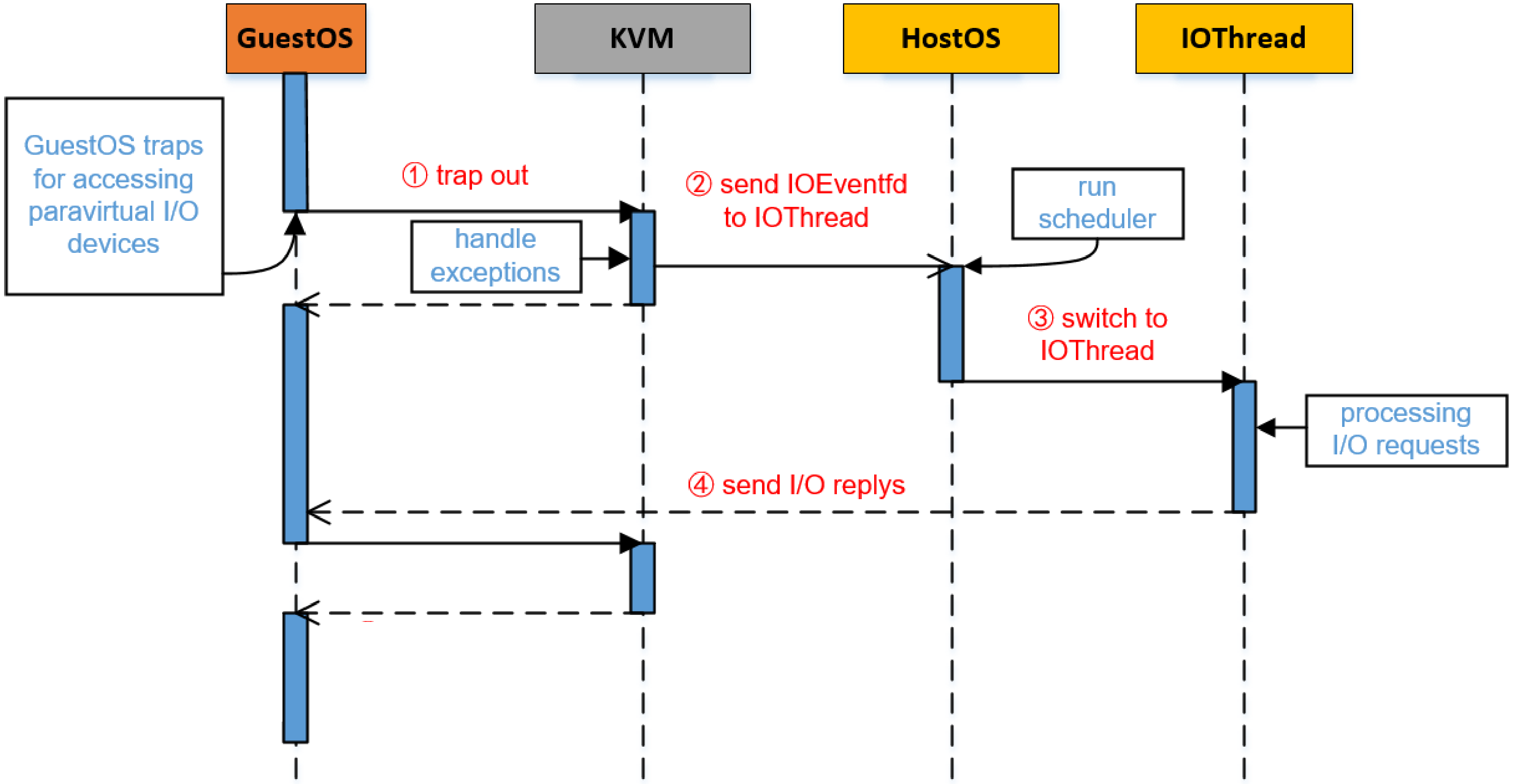
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



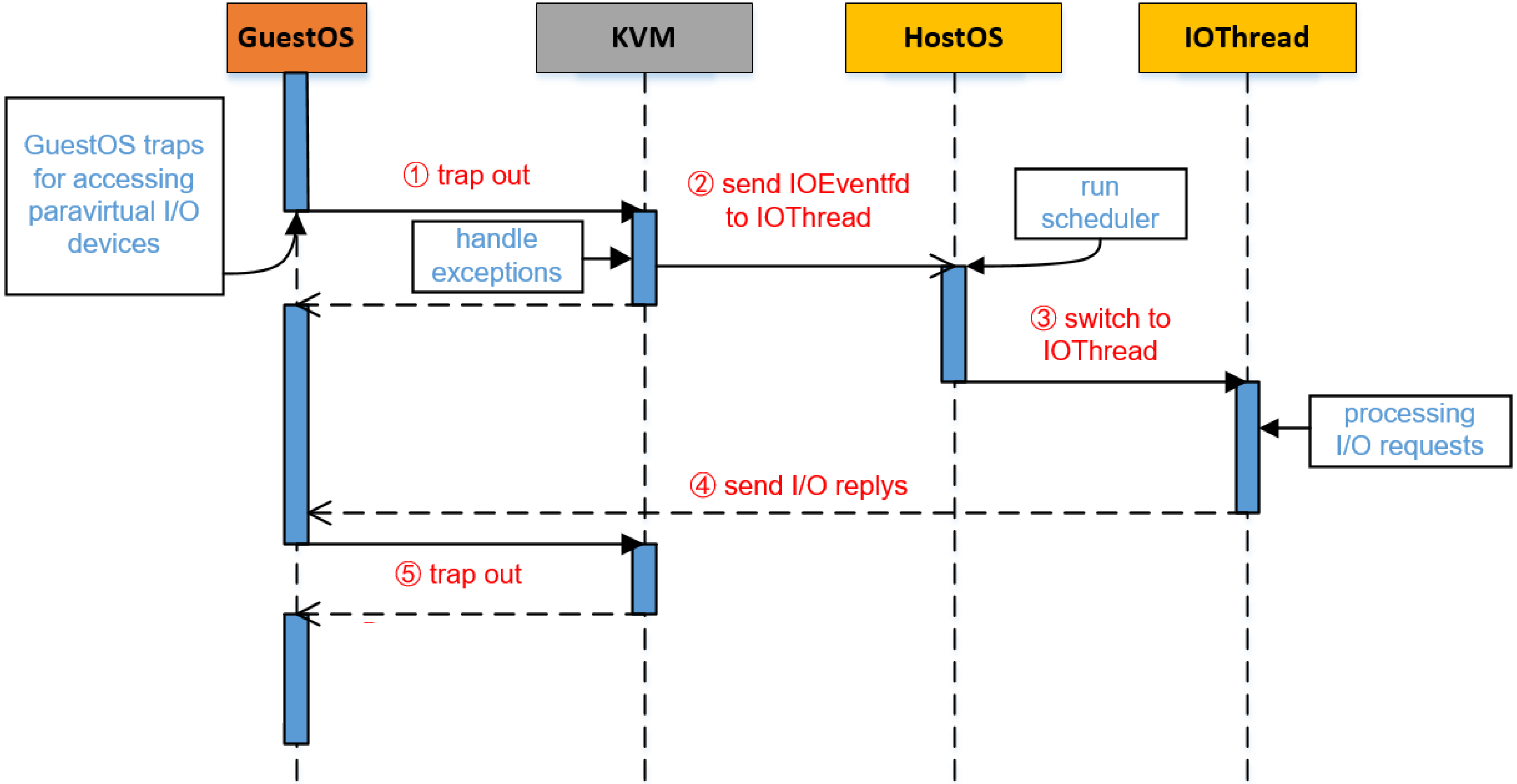
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



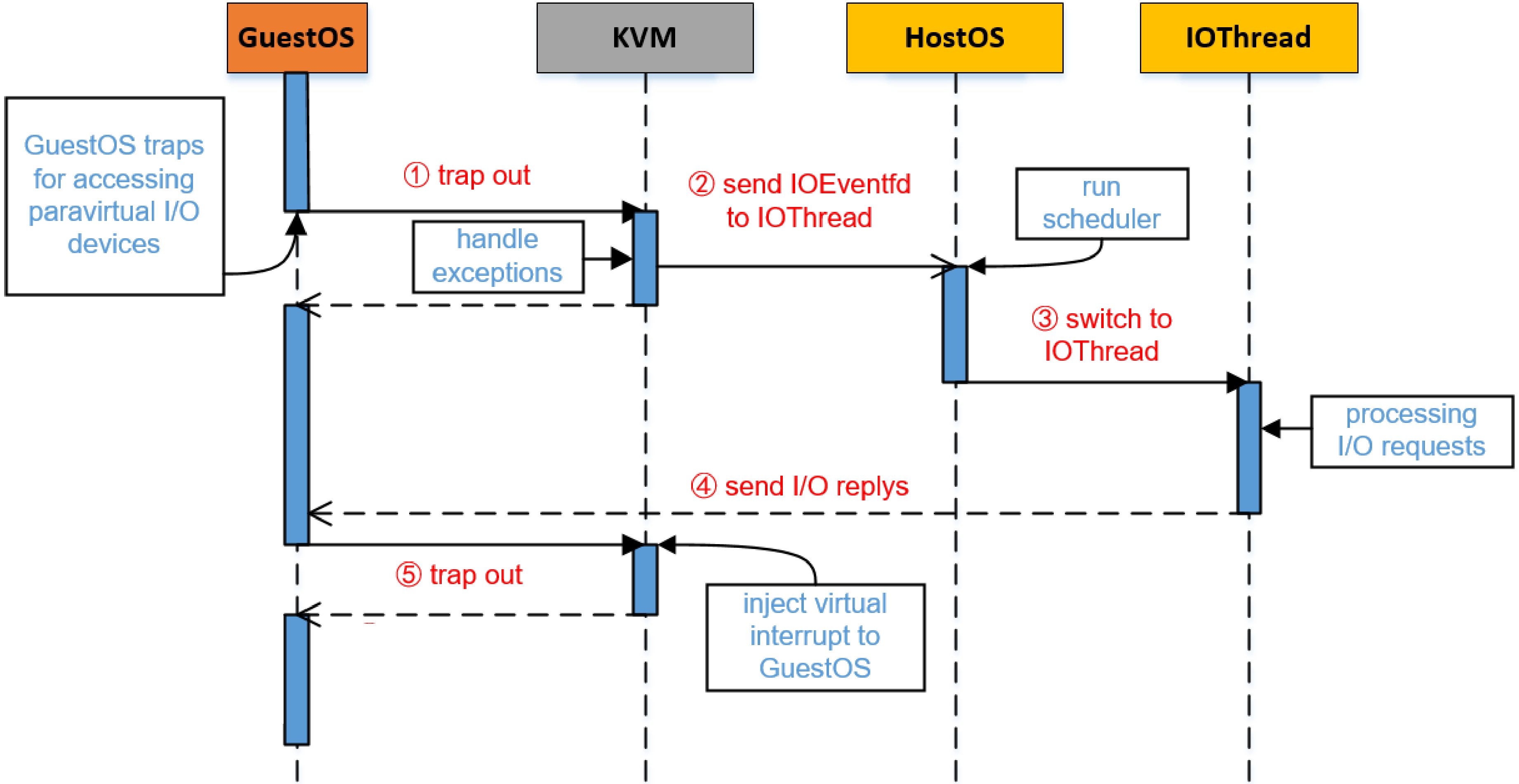
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



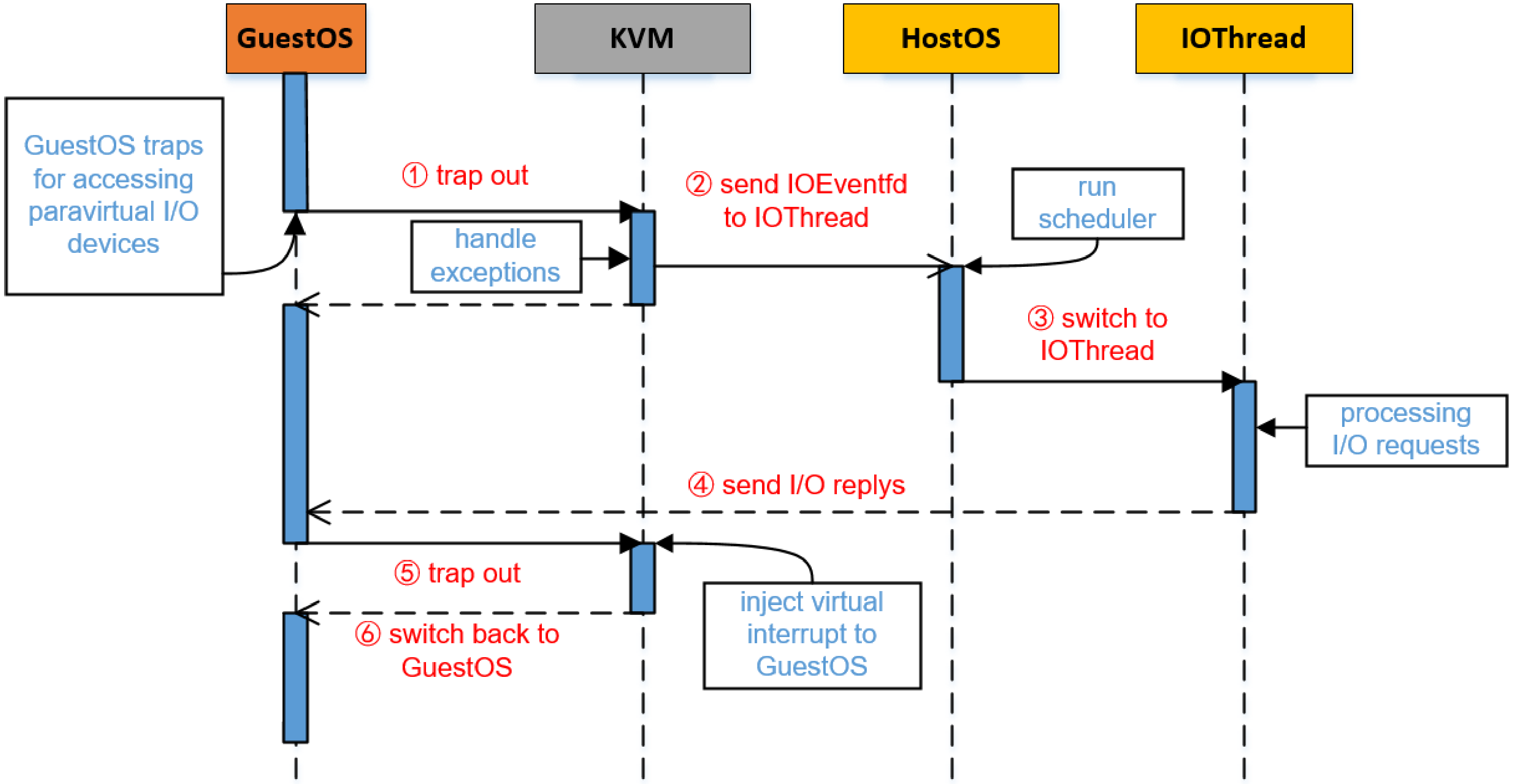
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



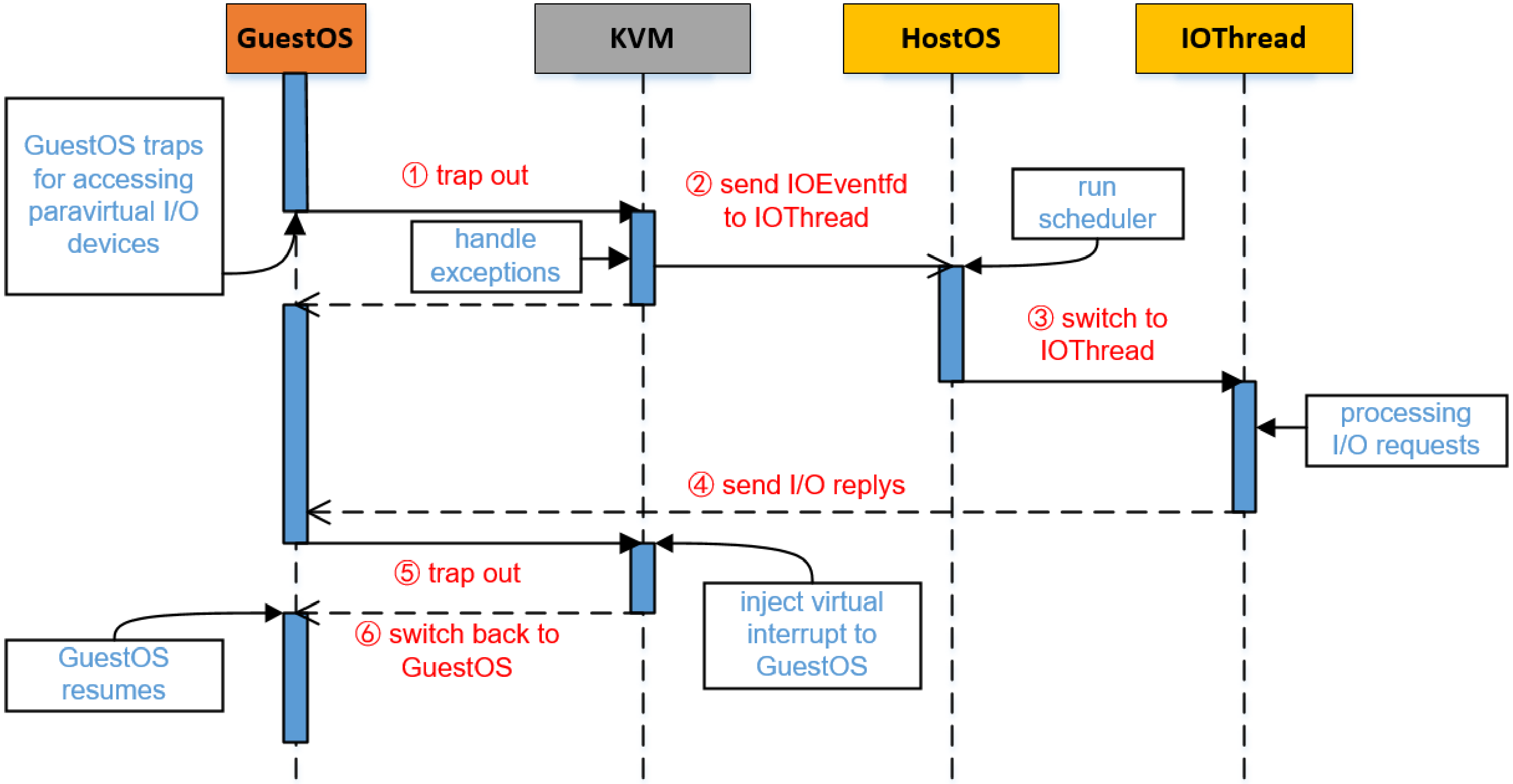
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



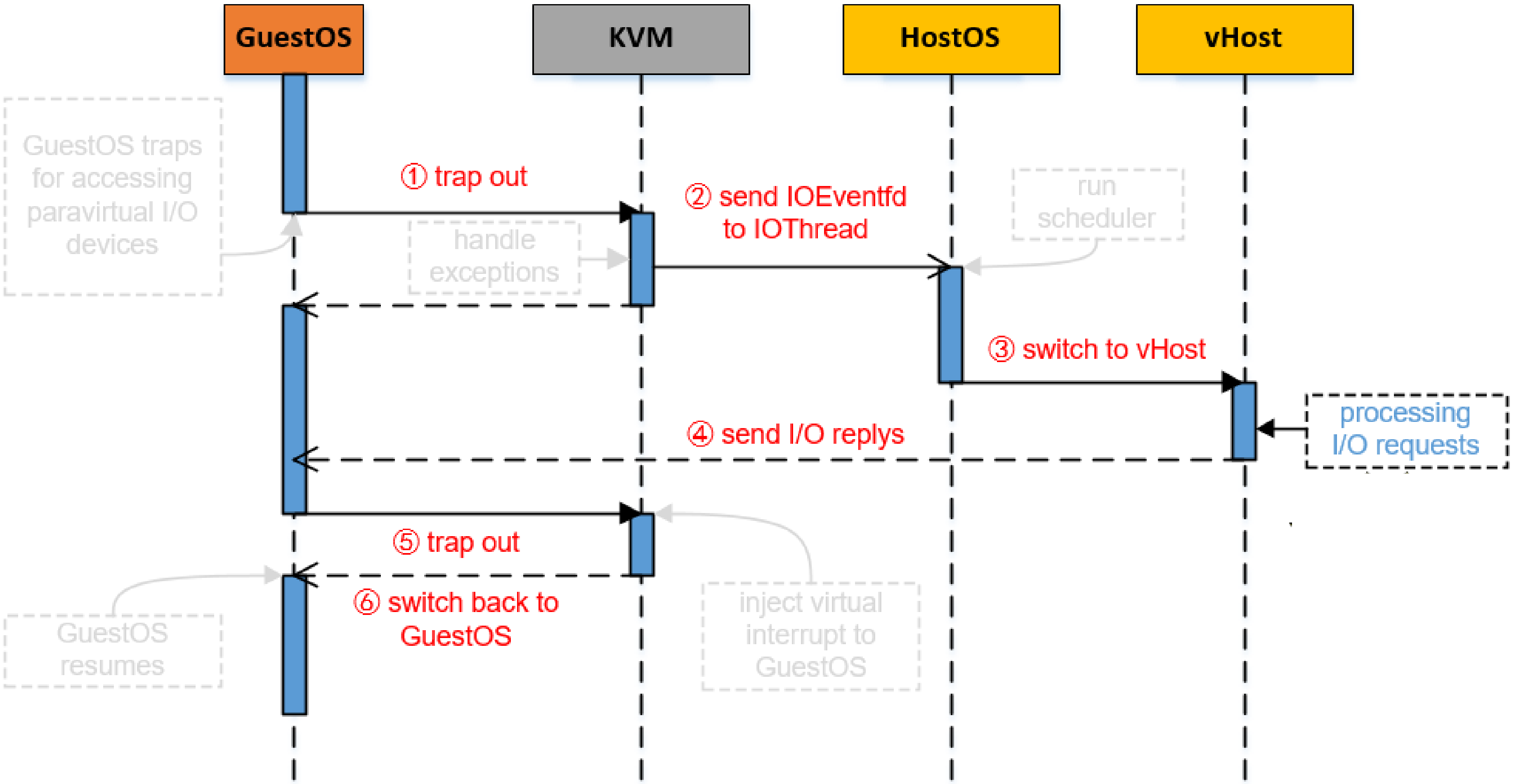
Motivation – Paravirtual I/O Devices

- VirtIO Based I/O Devices



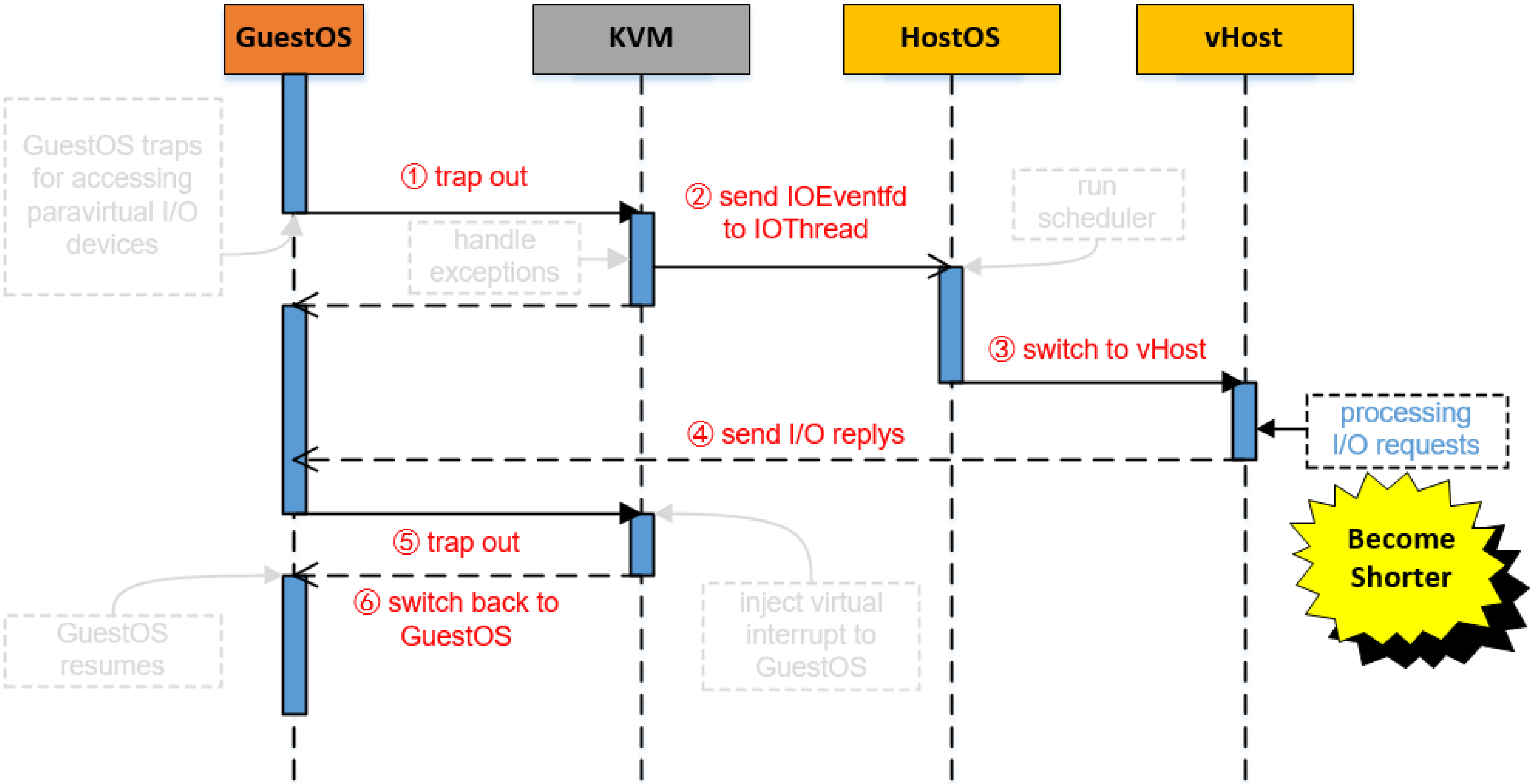
Motivation – Paravirtual I/O Devices

- vHost Based I/O Devices



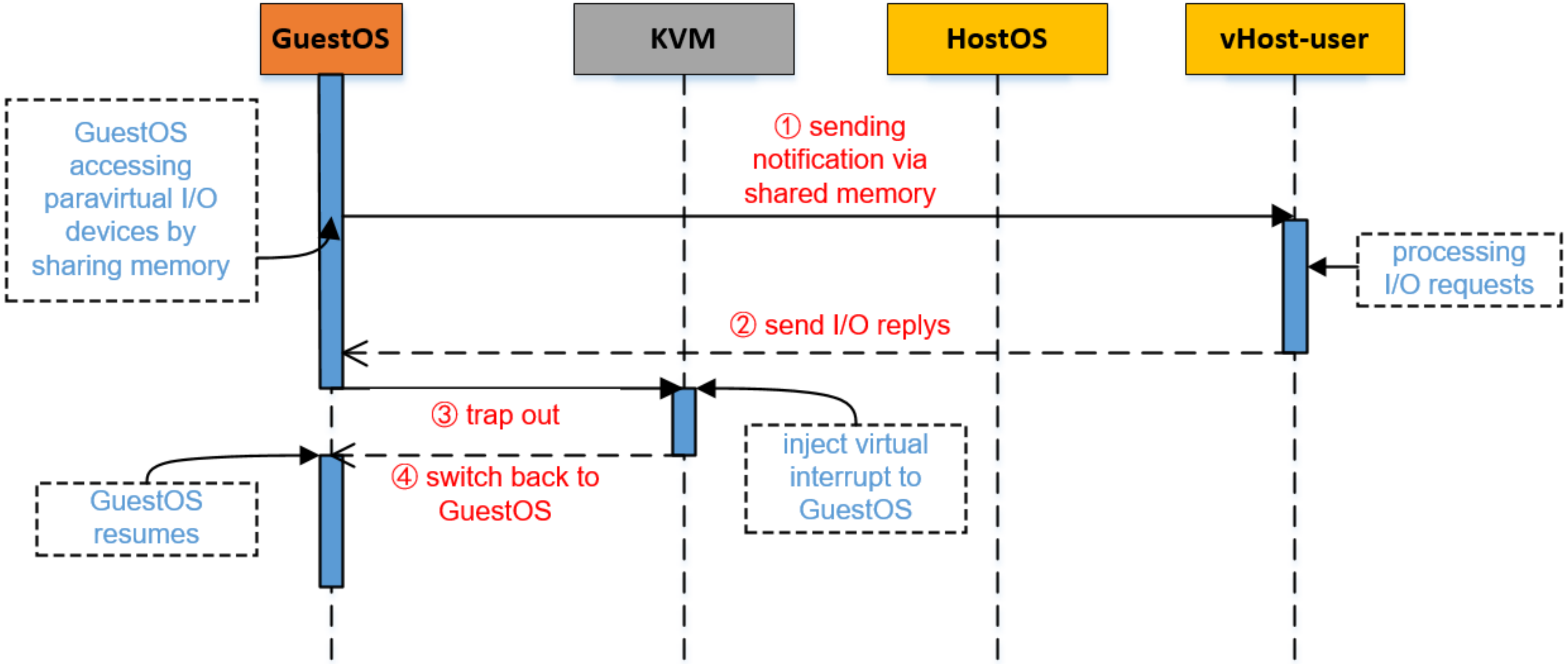
Motivation – Paravirtual I/O Devices

- vHost Based I/O Devices



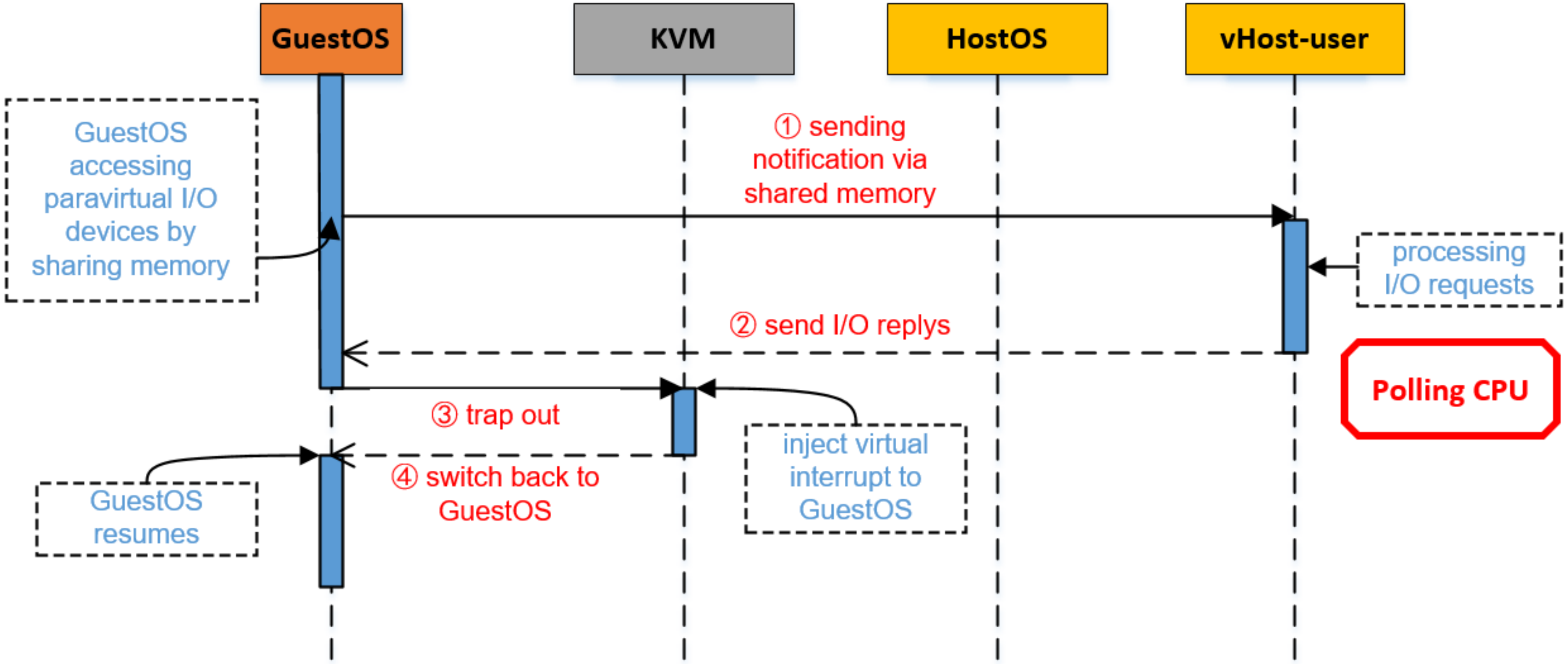
Motivation – Paravirtual I/O Devices

- vHost-user Based I/O Devices



Motivation – Paravirtual I/O Devices

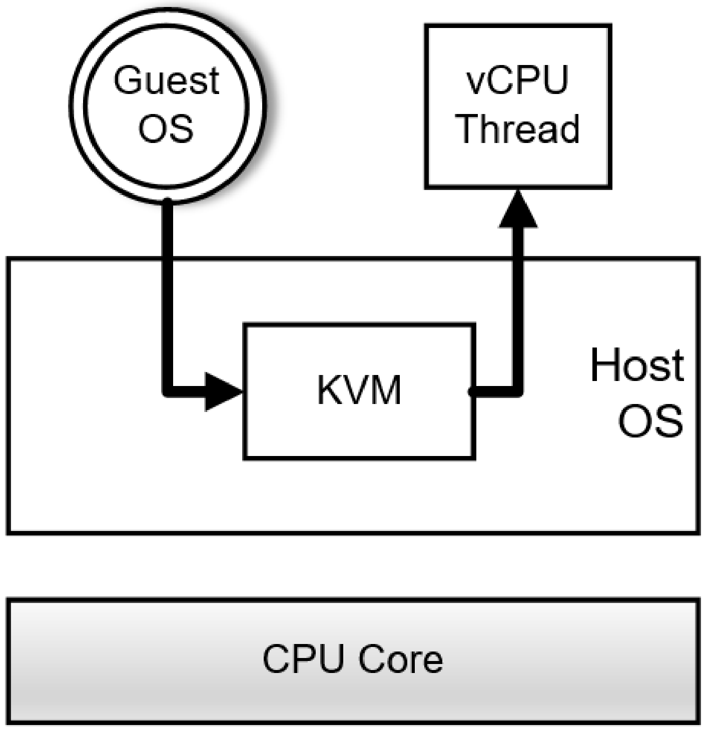
- vHost-user Based I/O Devices



Contents

Overview of HA-IOV Architecture

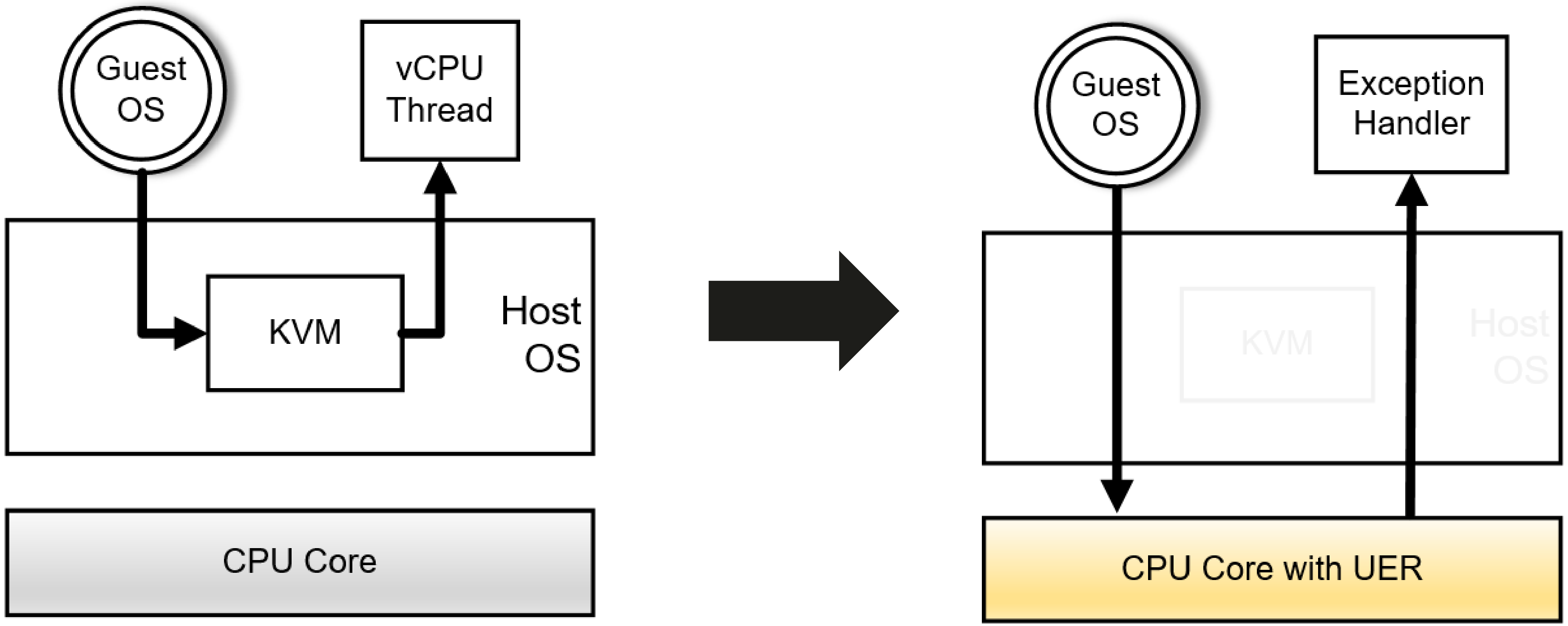
- HA-IOV for Full Emulated Virtual I/O Devices



◆ Costly context switch between Host and Guest, userspace and kernel.

Overview of HA-IOV Architecture

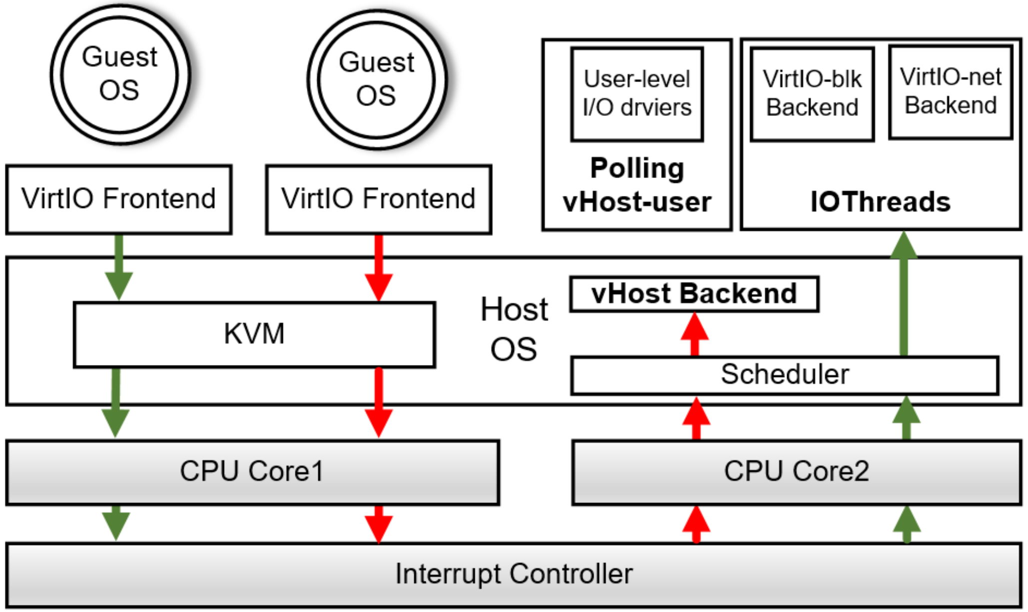
- HA-IOV for Full Emulated Virtual I/O Devices



- ◆ Costly context switch between Host and Guest, userspace and kernel.
- ◆ **User-level Exceptions Redirection(UER):** Alleviating context switch overheads by Directly delegating exceptions raised in Guest to be handled in Host user space bypassing KVM.

Overview of HA-IOV Architecture

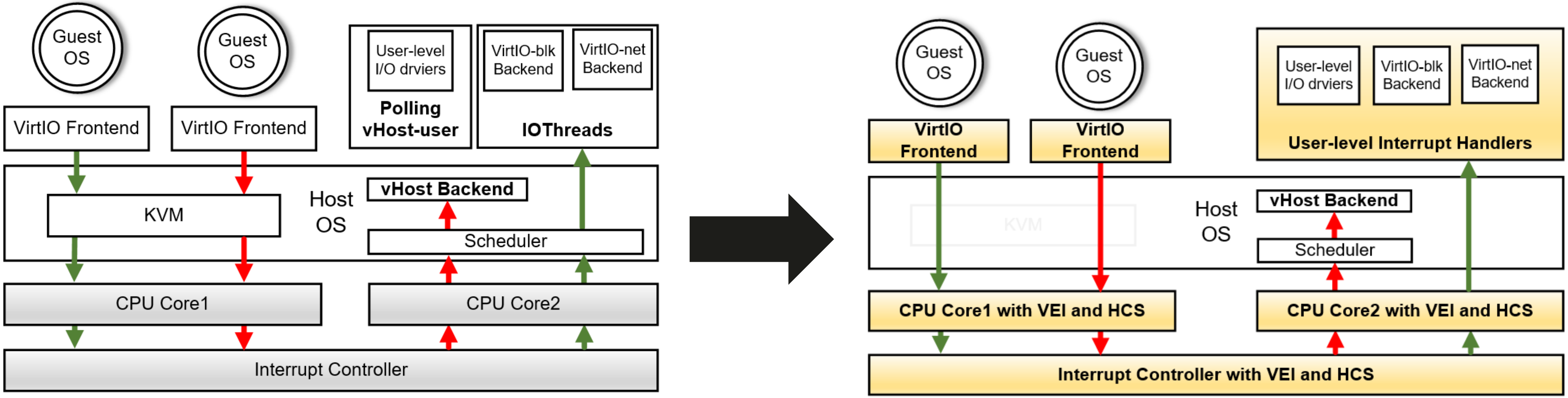
- HA-IOV for Paravirtual I/O Devices in Kernel and Userspace



- **VirtIO & vHost：** Guest traps out to send IPI causing context switch overheads
- **vHost-user：** polling threads prevent other threads running on the polling CPU cores to lower the CPU utilization

Overview of HA-IOV Architecture

- HA-IOV for Paravirtual I/O Devices in Kernel and Userspace

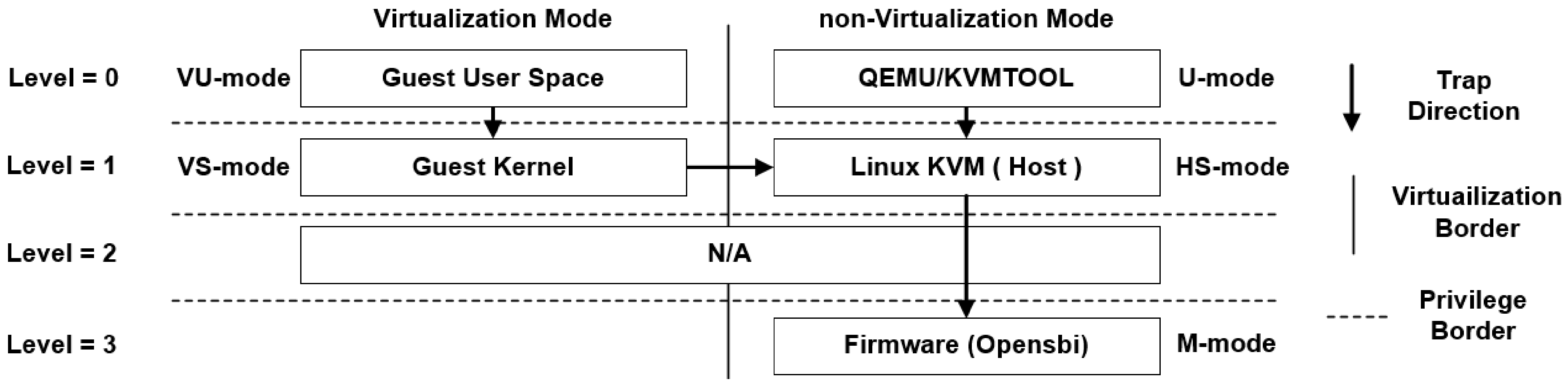


- **VirtIO & vHost：** Guest traps out to send IPI causing context switch overheads
- **vHost-user：** polling threads prevent other threads running on the polling CPU cores to lower the CPU utilization

- **Virtual Event Interrupt （VEI）：** Guest can send interrupts without trapping out to KVM.
- **Hardware-assisted Context Switch （HCS）：** user-level interrupt handlers can be woken up to handle interrupts in faster way bypassing kernel scheduler.

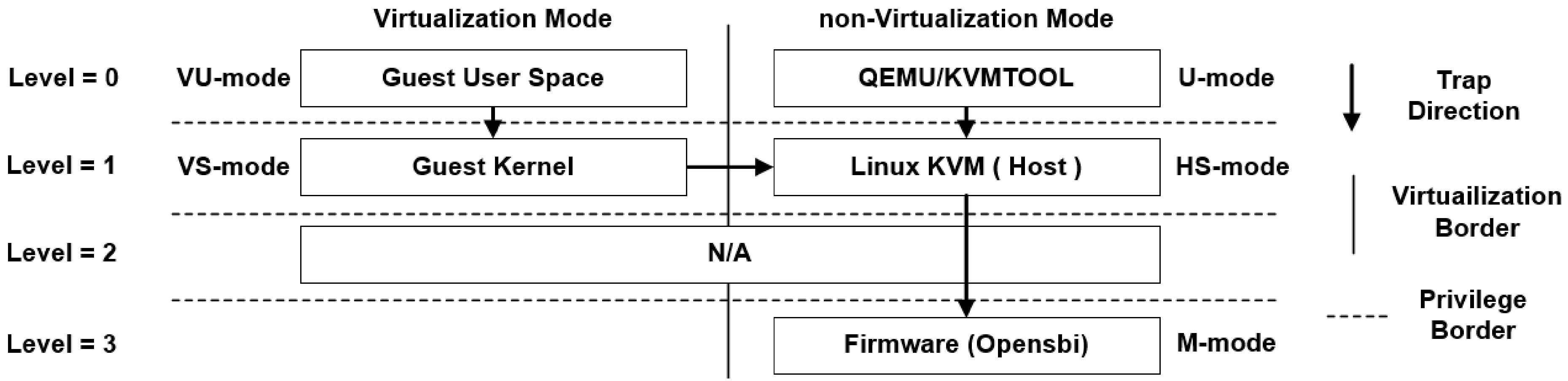
Implementation Prerequisite

- Privilege Mode In RISC-V Architecture



Implementation Prerequisite

- Privilege Mode In RISC-V Architecture



- N Extension
  - Adding RISC-V user-level interrupt and exception handler
  - Hardware can transfer control directly to a user-level trap handler without invoking the outer execution environment, such as KVM

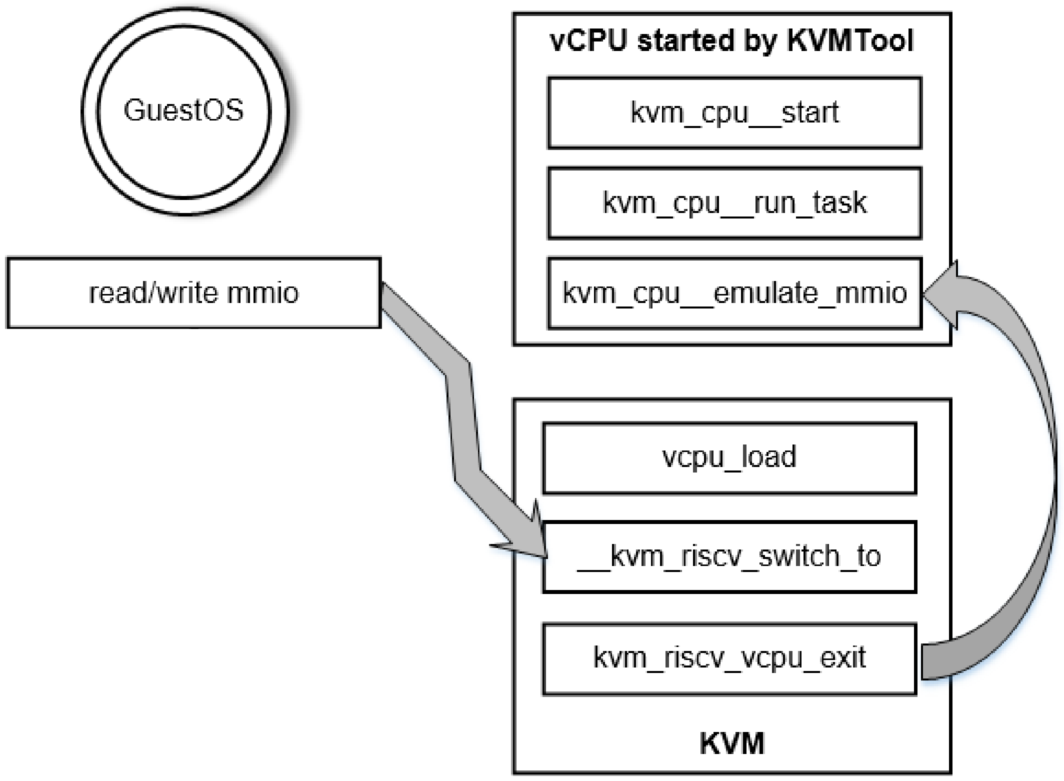
- Further Extending
  - Directly redirect exception occurs in VS-mode/VU-mode to U-mode
  - Interrupt VS-mode/VU-mode by user-level interrupts

Contents

- Background and Motivation
- Overview of HA-IOV Architecture
- HA-IOV Based Emulated Virtual I/O Devices
- HA-IOV Based Kernel Paravirtual I/O Devices
- HA-IOV Based Userspace Paravirtual I/O Devices
- Conclusion and Future Work

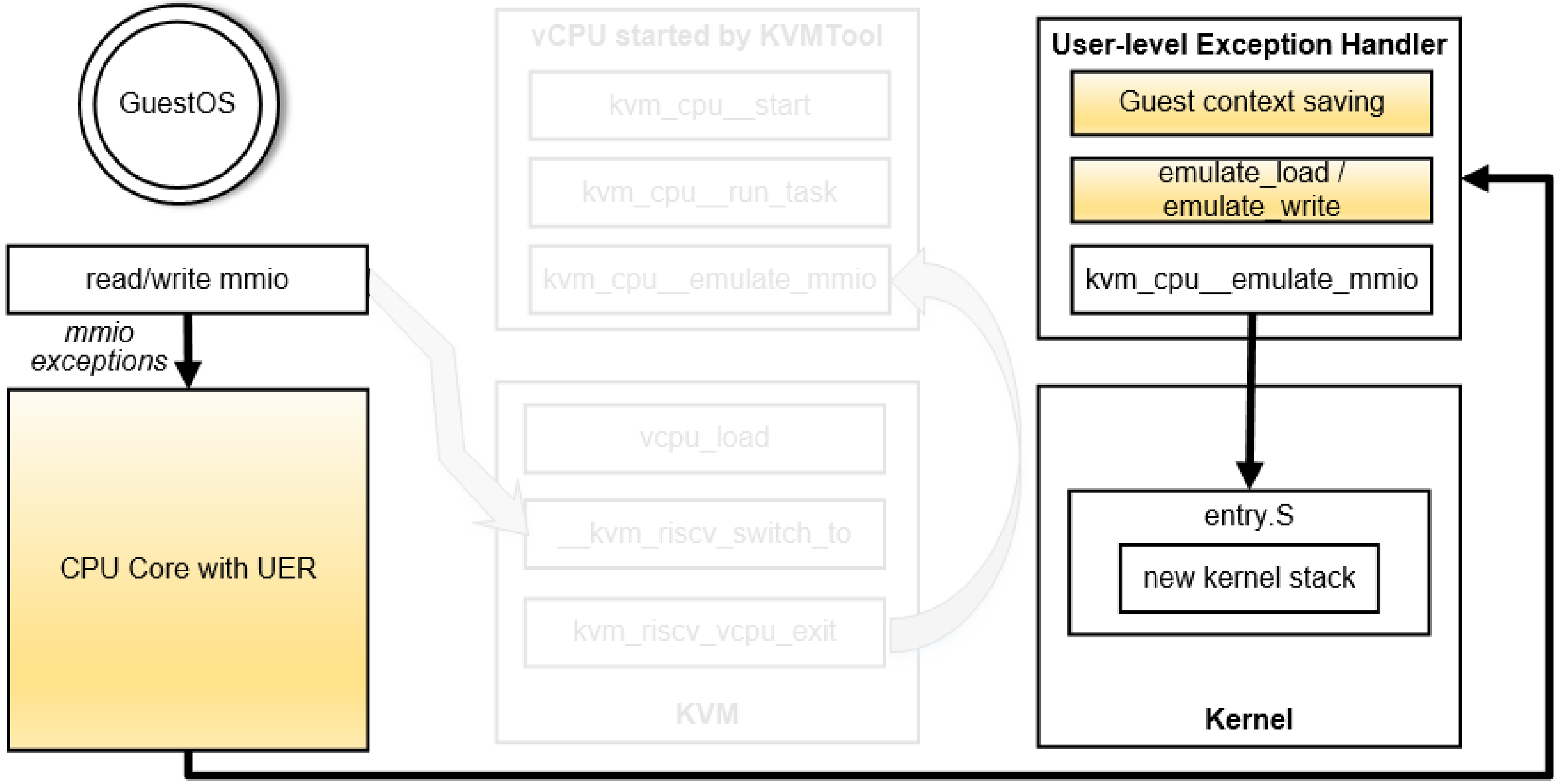
HA-IOV for Emulated Virtual I/O Devices

- Overview



HA-IOV for Emulated Virtual I/O Devices

- Overview



HA-IOV for Emulated Virtual I/O Devices

- Extended User-level Exception Redirection Mechanism
  - Reusing CSRs of N extension

Registers	Description
ustatus	Keeping track of and controls the hart’s current **operating state in userspace**
uscratch	The **pointer of data structure** of user-level exception handler
uepc	The **instruction** that raises the user-level exception
ucause	The **reason** to cause the user-level exception
utvec	The **entry address** of handling user-level exceptions
utval	The **related value** the upec operates on, such mmio address

HA-IOV for Emulated Virtual I/O Devices

- Extended User-level Exception Redirection Mechanism
  - Reusing CSRs of N extension
  - Delegating the exception in Guest to user-level trap handler.

Delegation	Types	Modification	Description
huedeleg	Register	Added	Hypervisor user exception delegation register
ustatus	Register	Extended	Add two fields, called UPV and UPP, in ustatus
URET	Instruction	Extended	Allow to return to VU-mode/VS-mode from U-mode
hstatus	Register	Extended	Add a field, called HUR, in hstatus

HA-IOV for Emulated Virtual I/O Devices

- Extended User-level Exception Redirection Mechanism
  - Reusing CSRs of N extension
  - Delegating the exception in Guest to user-level trap handler.
  - Two MMIO page fault exceptions are added.

Delegation	Types	Modification	Description
huedeleg	Register	Added	Hypervisor user exception delegation register
ustatus	Register	Extended	Add two fields, called UPV and UPP, in ustatus
URET	Instruction	Extended	Allow to return to VU-mode/VS-mode from U-mode
hstatus	Register	Extended	Add a field, called HUR, in hstatus
Exceptions	**Types**	**Modification**	**Description**
Load MMIO page fault	Trap Code	Added	Page fault caused by write MMIO in Guest
Store MMIO page fault	Trap Code	Added	Page fault caused by read MMIO in Guest
MMIO field	PTE Format	Extended	Add a field, called MMIO, in PTE

HA-IOV for Emulated Virtual I/O Devices

- Evaluation

  - Environment

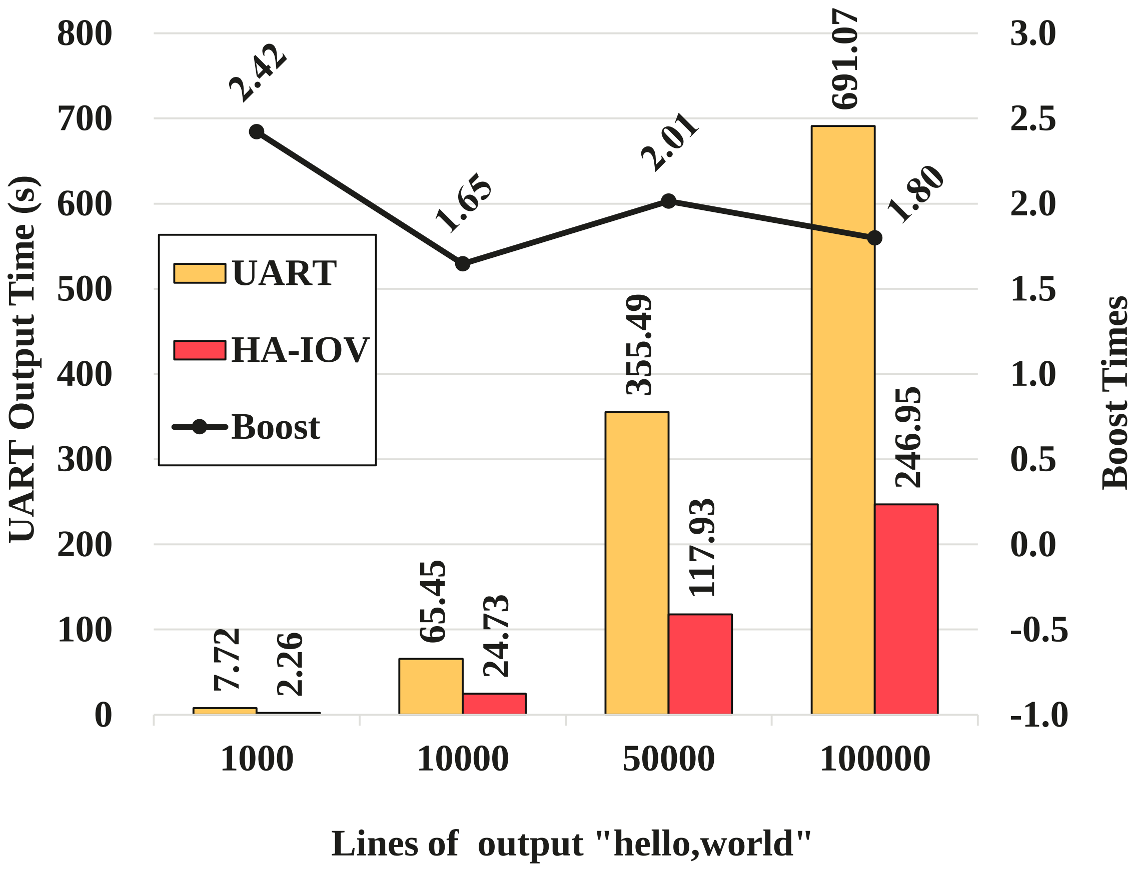
    1. Hisilicon Kunpeng 920 2600MHz
    2. RISC-V QEMU Emulator V0.5.1
    3. RISC-V KVMTOOL V1
    4. RISC-V KVM V10
    5. RISC-V HostOS with 4 CPU 2048M
    6. RISC-V GuestOS with 1 CPU 1024M

  - Experiments

    Output 1K, 10K, 50K, 100K lines of "hello,world" to stdout (terminal)

  - Results

    HA-IOV achieves nearly 2X faster than the original one (the Lower the better)

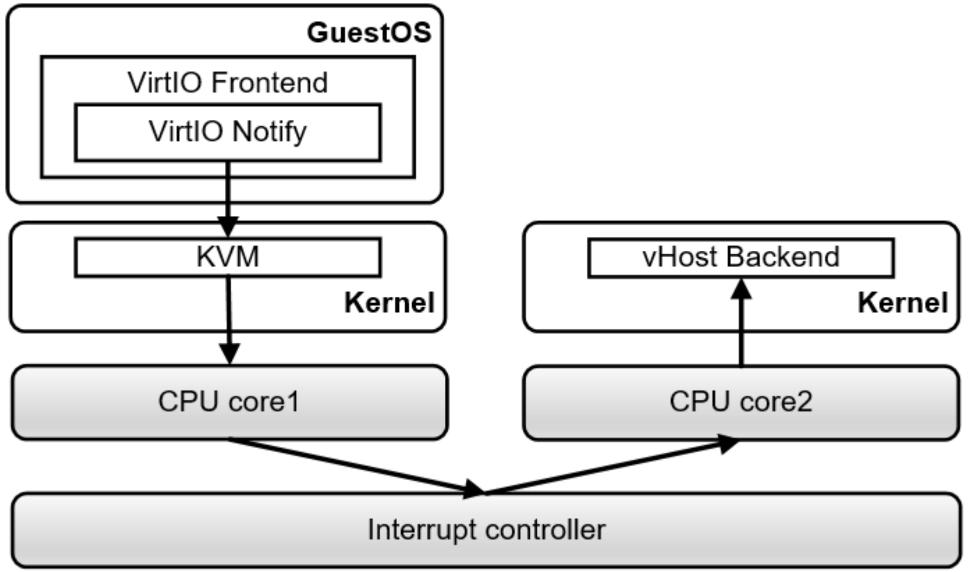


Contents

- Background and Motivation
- Overview of HA-IOV Architecture
- HA-IOV Based Emulated Virtual I/O Devices
- HA-IOV Based Kernel Paravirtual I/O Devices
- HA-IOV Based Userspace Paravirtual I/O Devices
- Conclusion and Future Work

HA-IOV for Paravirtual I/O Devices in Kernel

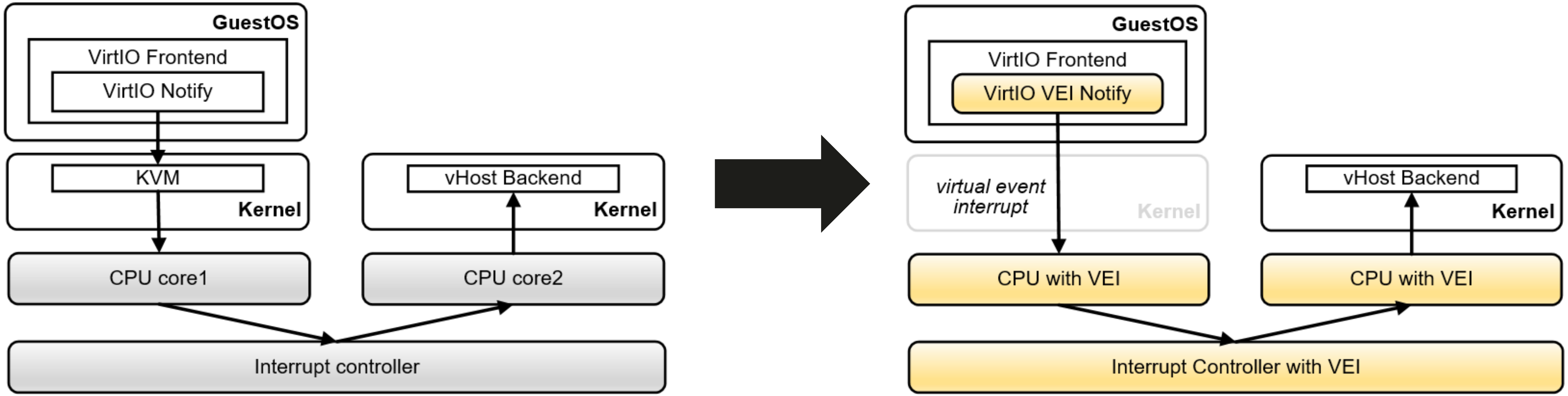
- Overview



◆ **vHost:** Guest traps out to send IPI causing context switch overheads

HA-IOV for Paravirtual I/O Devices in Kernel

- Overview

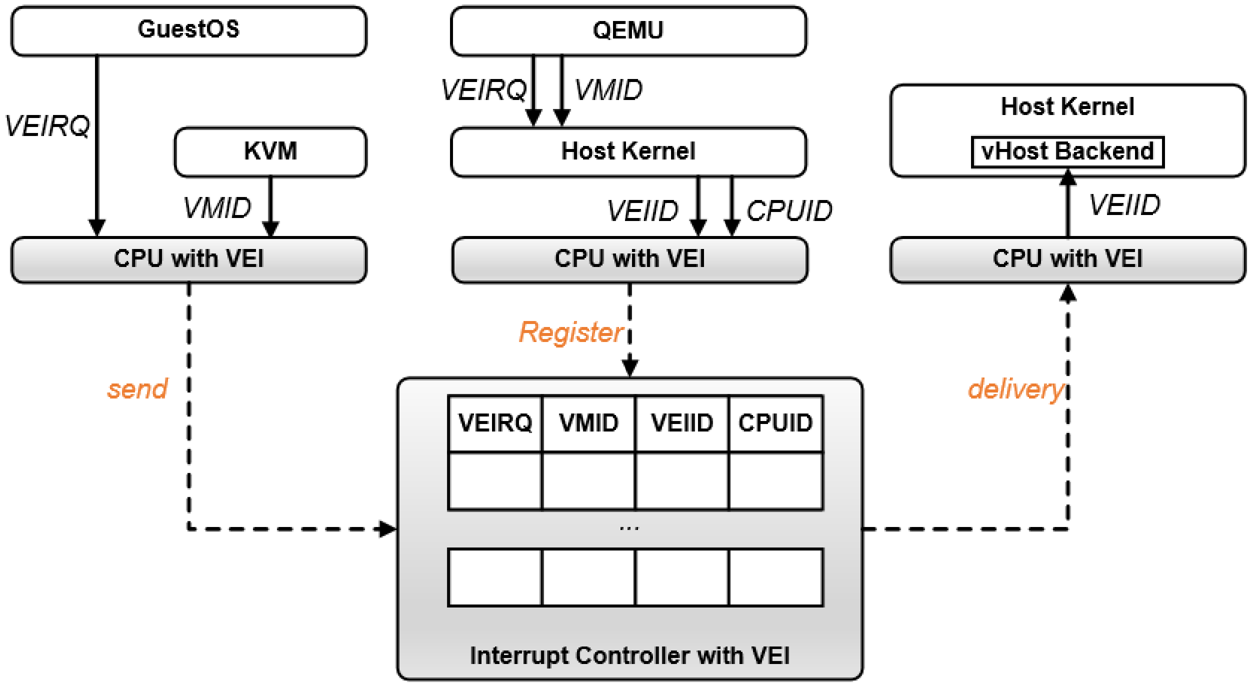


◆ **vHost:** Guest traps out to send IPI causing context switch overheads

◆ **Virtual Event Interrupt (VEI) :** allowing Guest to send supervisor interrupt without existing to notify Host kernel threads in target CPU

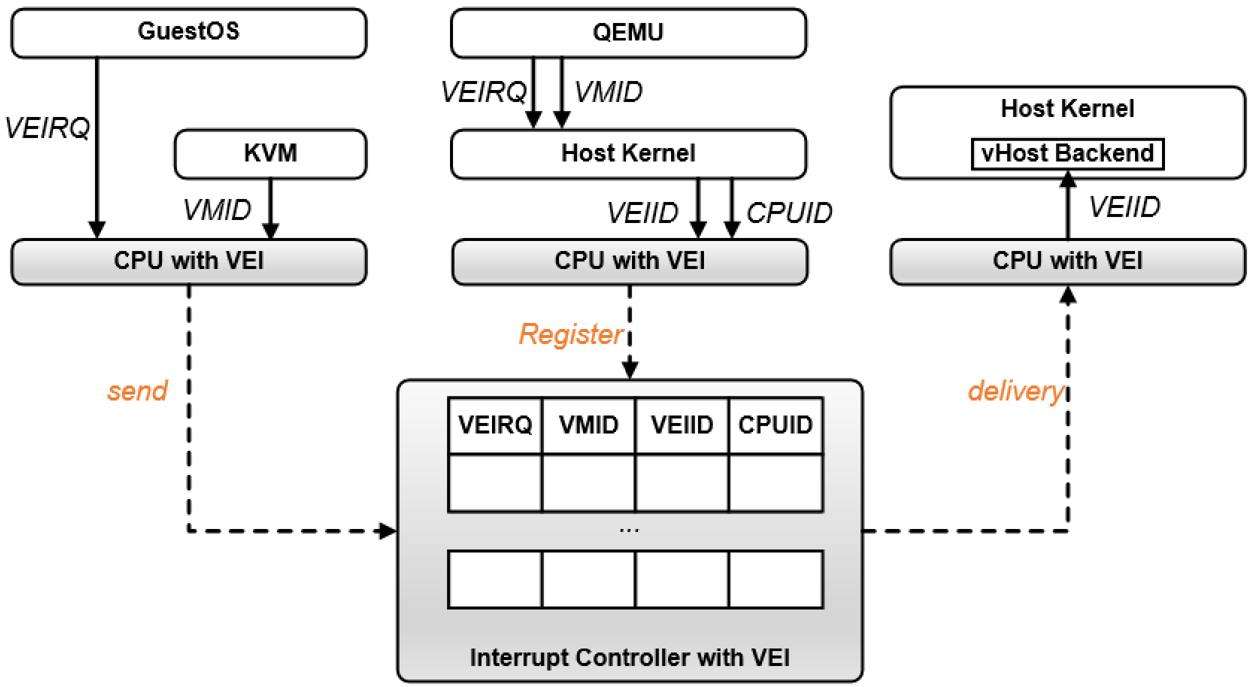
HA-IOV for Paravirtual I/O Devices in Kernel

- Supervisor Virtual Event Interrupts
  - Mapping and Registration Information
    - ✓ VEIID (VEI Identity, or VEI Physical Number)
    - ✓ VEIRQ (VEI Request Number)
    - ✓ Qemu provides VEIRQ and VMID
    - ✓ Kernel provides VEIID and CPUID
    - ✓ VEIRQ+VMID ←→ VEIID+CPUID



HA-IOV for Paravirtual I/O Devices in Kernel

- Supervisor Virtual Event Interrupts
  - Mapping and Registration Information
    - ✓ VEIID (VEI Identity, or VEI Physical Number)
    - ✓ VEIRQ (VEI Request Number)
    - ✓ Qemu provides VEIRQ and VMID
    - ✓ Kernel provides VEIID and CPUID
    - ✓ VEIRQ+VMID ←→ VEIID+CPUID
  - Routing Steps
    1. Guest sends VEI by writing VEIRQ to a new added CPU register. The register is allowed to access in Guest, so there is no need for Guest to trap out.
    2. VMID is obtained by VEI module in CPU to further query the registered mapping information presented in the interrupt controller.
    3. When the target CPUID is found, a physical interrupt identified by VEIID is delivered to the Host kernel threads in target CPU.

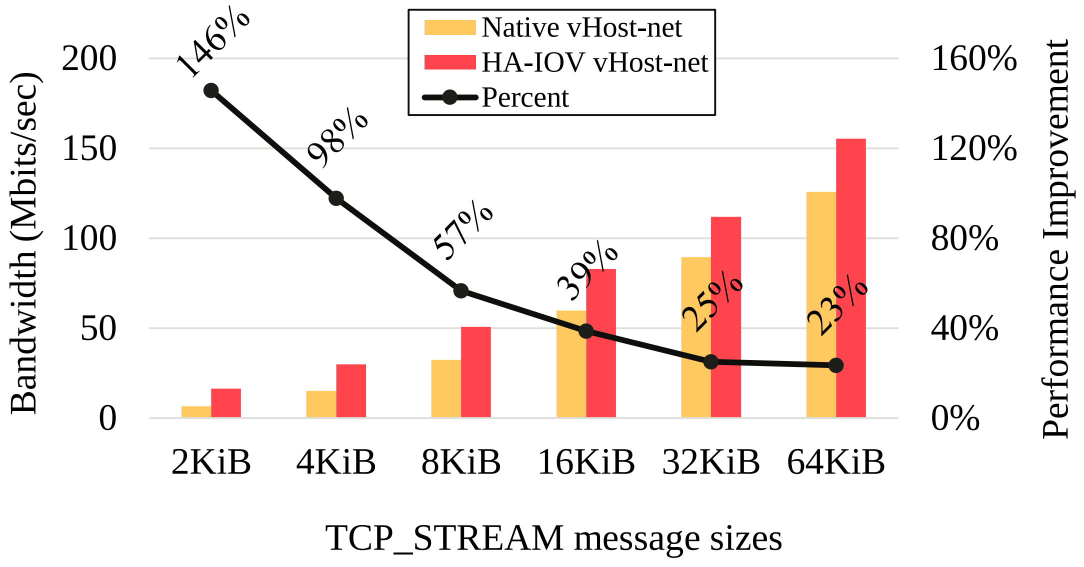


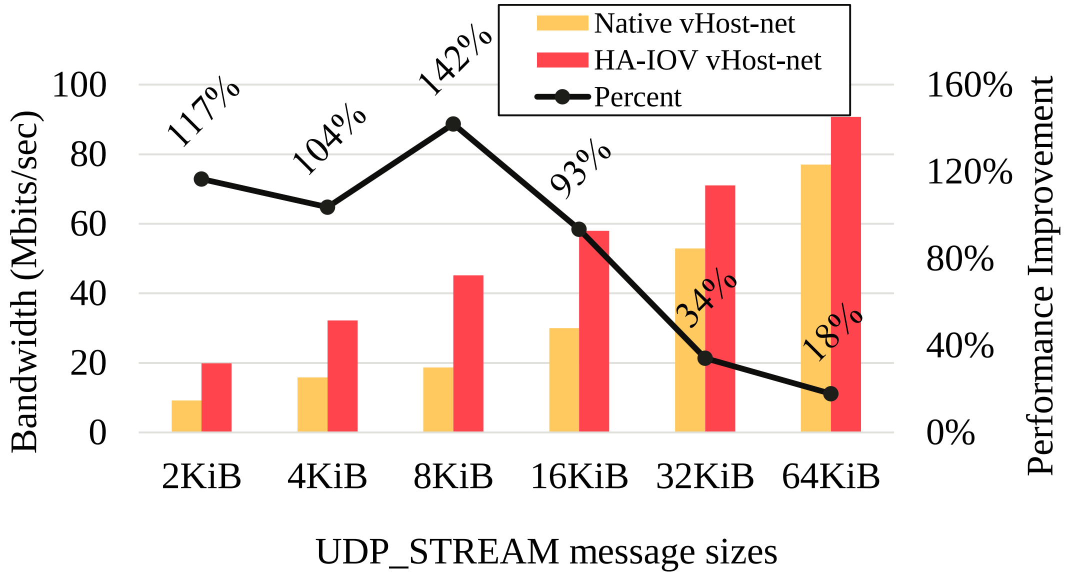
HA-IOV for Paravirtual I/O Devices in Kernel

- Evaluation
  - Environment
    1. Hisilicon Kunpeng 920 2600MHz
    2. RISC-V QEMU Emulator V0.5.1
    3. RISC-V QEMU V0.5.1
    4. RISC-V KVM V10
    5. RISC-V HostOS with 4 CPU 2048M
    6. RISC-V GuestOS with 1 CPU 1024M
  - Experiment

    NetPerf configured with UDP and TCP
  - Results

    Performance of HA-IOV based vHost-net increase over 100% when message sizes are small (the higher the better)

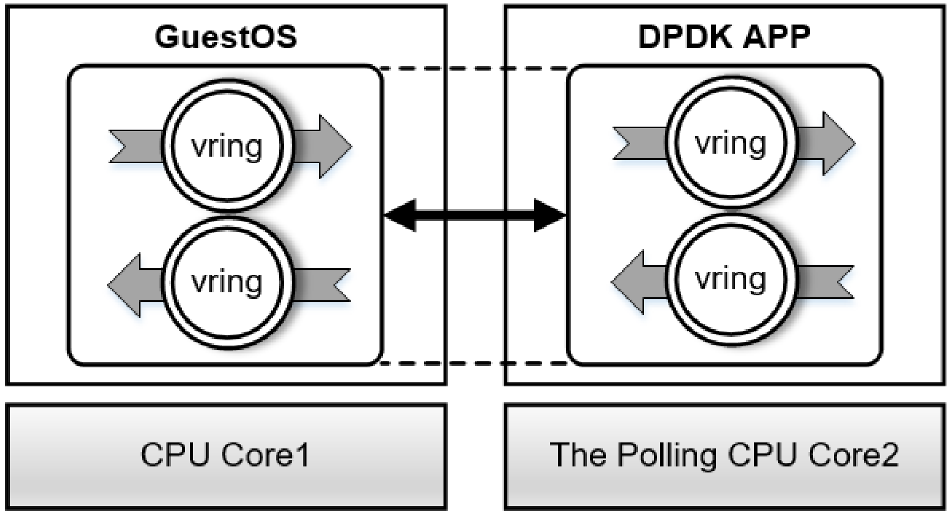




Contents

HA-IOV for Paravirtual I/O Devices in Userspace

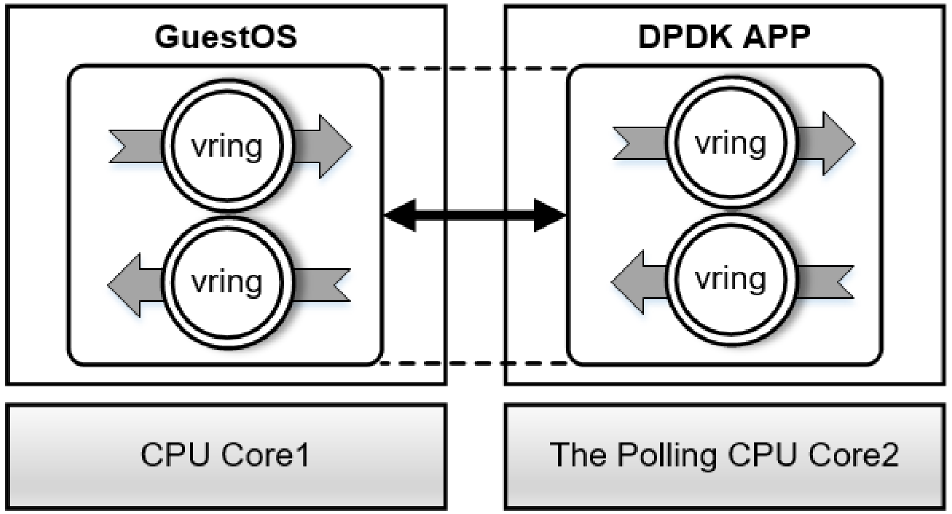
- Overview

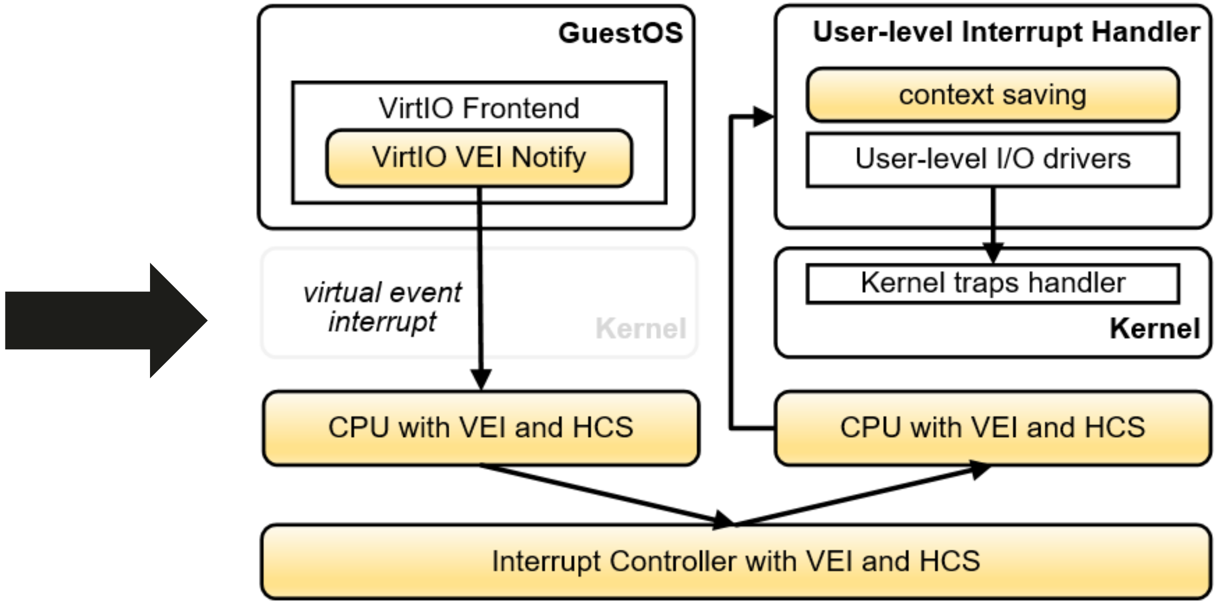


- **Sharing Memory** : GuestOS interacts with user-level I/O devices via shared memory
- **Polling mode** : VirtIO backend implements as polling threads, which keep other threads running on the polling CPU core.

HA-IOV for Paravirtual I/O Devices in Userspace

- Overview





- **Sharing Memory** : GuestOS interacts with user-level I/O devices via shared memory
- **Polling mode** : VirtIO backend implements as polling threads, which keep other threads running on the polling CPU core.

- **User-level VEI:** VirtIO backend are implemented as user-level interrupt handlers, which are triggered by user-level interrupts.
- **Hardware-assisted context switch (HCS) :** Swapping the memory space and data structure for interrupt handlers.

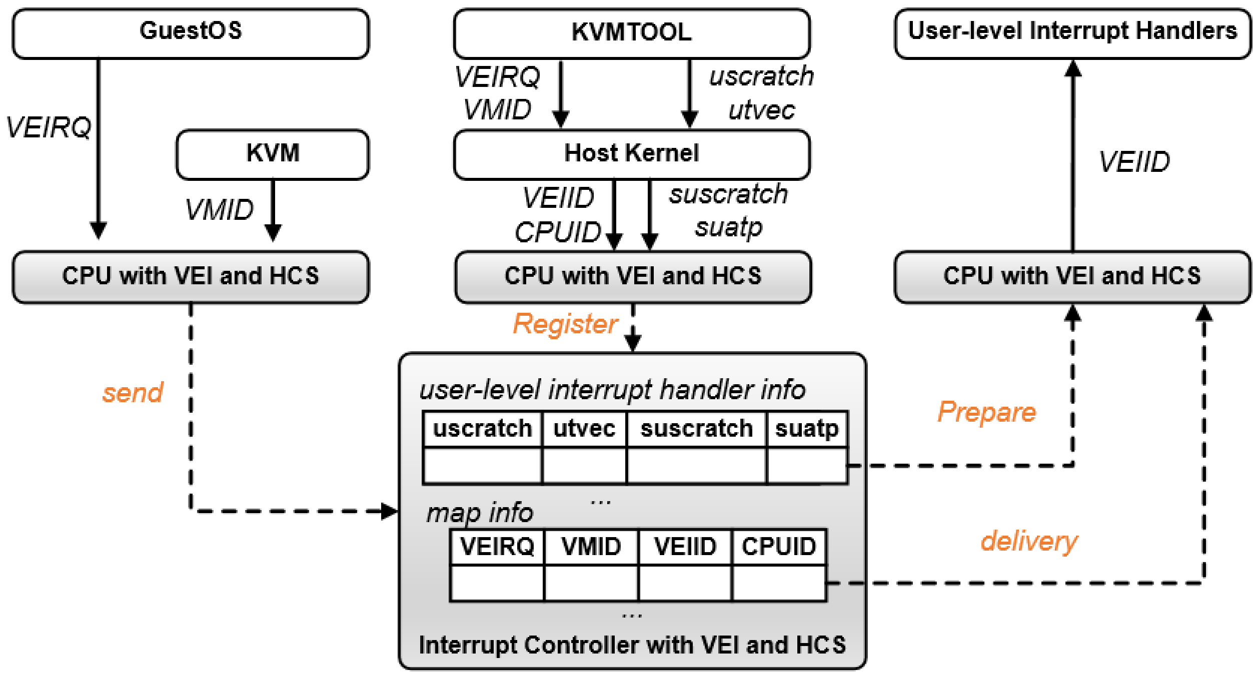
HA-IOV for Paravirtual I/O Devices in Userspace

- User-level Virtual Event Interrupt
  - Register Information

    uscratch , utvec,

    suscratch (kernel data structure),

    suatp (memory space)



HA-IOV for Paravirtual I/O Devices in Userspace

- User-level Virtual Event Interrupt
  - ➢ Register Information

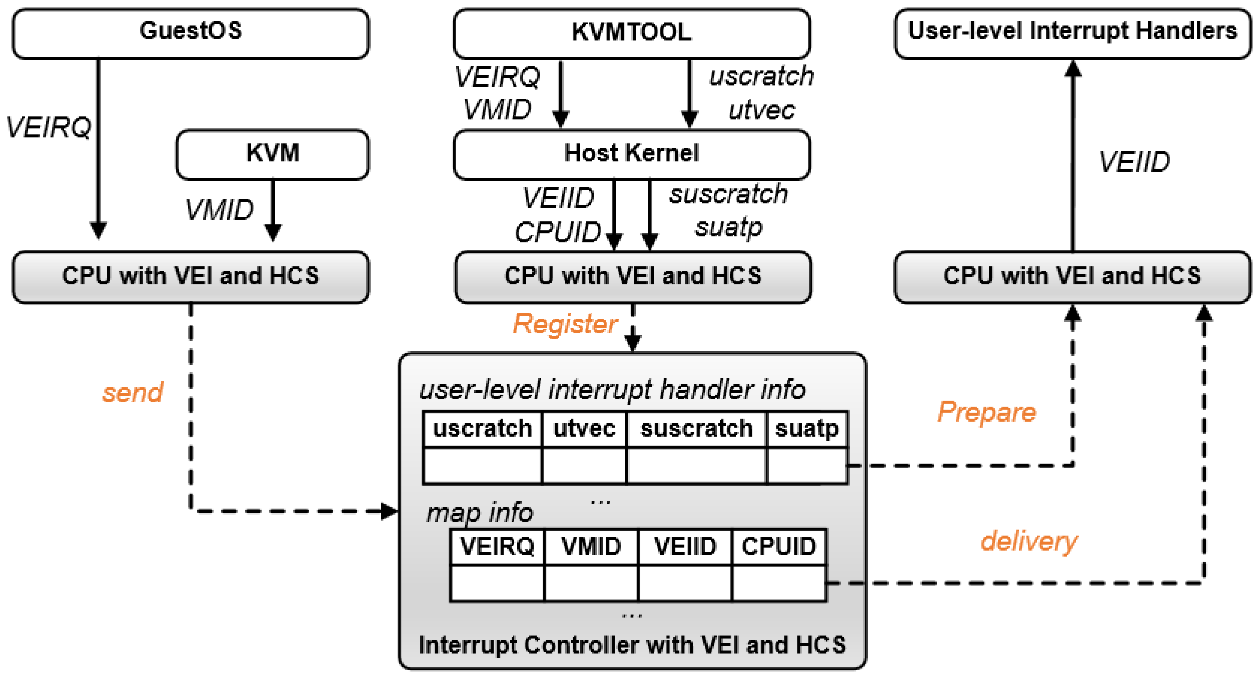
    uscratch , utvec,

    suscratch (kernel data structure),

    suatp (memory space)

  - ➢ Prepare Context for interrupt handlers

    <uscratch, utvec, suscratch, suatp> (IC) →
    <uscratch, utvec, suscratch, suatp> (CPU)



HA-IOV for Paravirtual I/O Devices in Userspace

- User-level Virtual Event Interrupt
  - ➢ Register Information

    uscratch , utvec,

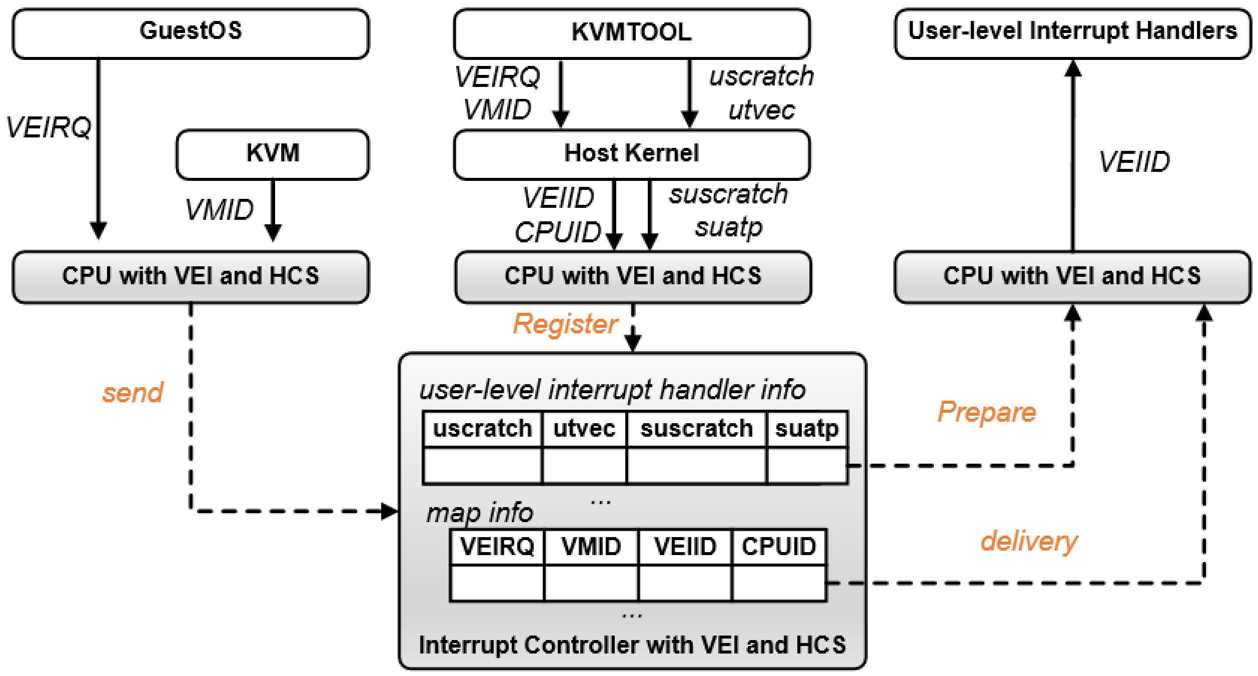
    suscratch (kernel data structure),

    suatp (memory space)
  - ➢ Prepare Context for interrupt handlers

    <uscratch, utvec, suscratch, suatp> (IC) →
    <uscratch, utvec, suscratch, suatp> (CPU)
  - ➢ When handling a user-level interrupt

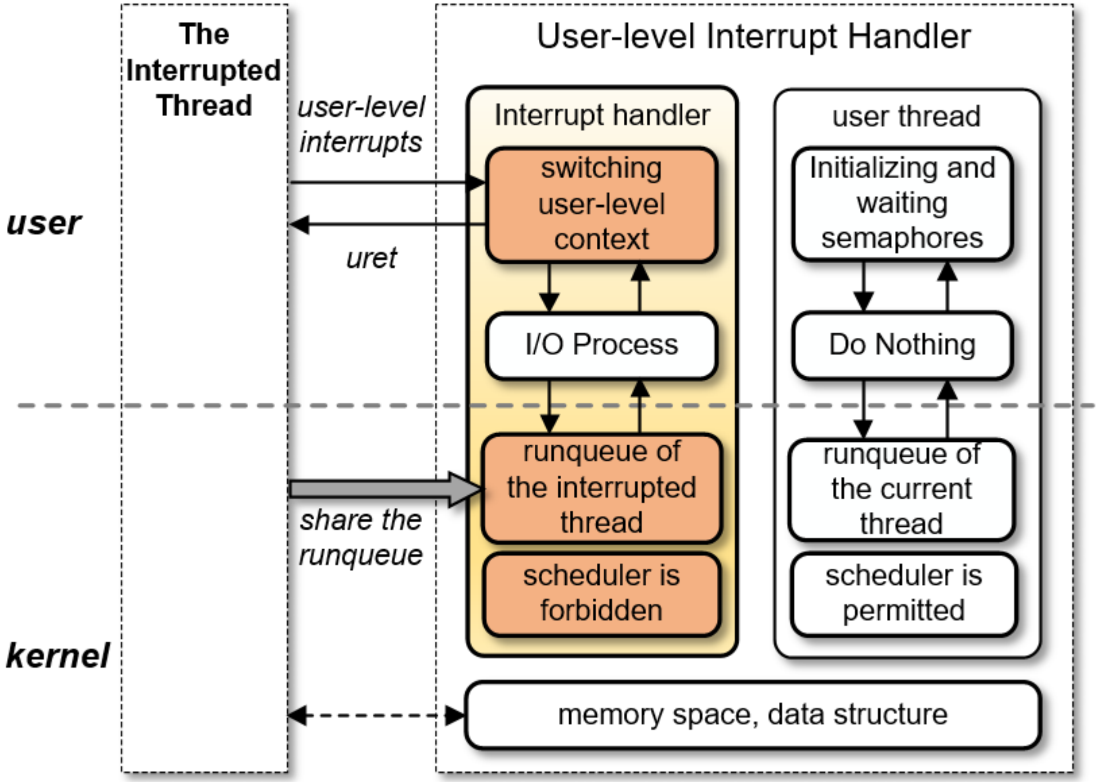
    <suscratch, suatp> (CPU) ←→

    <sscratch, satp> (CPU)



HA-IOV for Paravirtual I/O Devices in Userspace

- User-level Interrupt Handler
  - Consisting of a **interrupt handler** and a **user thread**
    1. Both sharing same *memory space* and *kernel data structure*
    2. The interrupt handler runs by HCS
    3. The user thread runs by kernel scheduler.
    4. Scheduling in the interrupt handler is disabled, while the user thread can be scheduled out



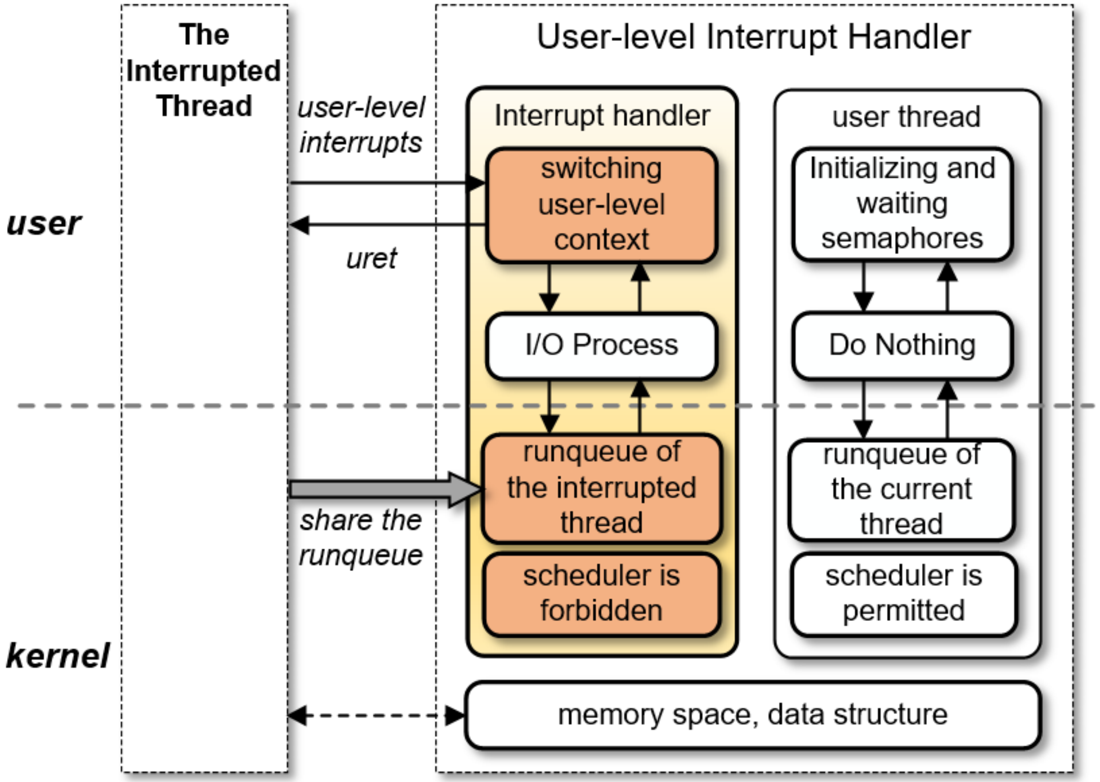
HA-IOV for Paravirtual I/O Devices in Userspace

- User-level Interrupt Handler
  - ➢ Consisting of a **interrupt handler** and a **user thread**
    1. Both sharing same *memory space* and *kernel data structure*
    2. The interrupt handler runs by HCS
    3. The user thread runs by kernel scheduler.
    4. Scheduling in the interrupt handler is disabled, while the user thread can be scheduled out
  - ➢ Priority of VS-mode/VU-mode is defined to be less than U-mode

    Running VMs is able to be interrupted by user-level interrupts for quickly handling user-level interrupts.
  - ➢ If the target CPU core is in S-mode

    The user thread will be scheduled by door bell interrupt raised by user-level vei

    The interrupt handler is then triggered by the user-level vei to run I/O process by interrupting the user thread

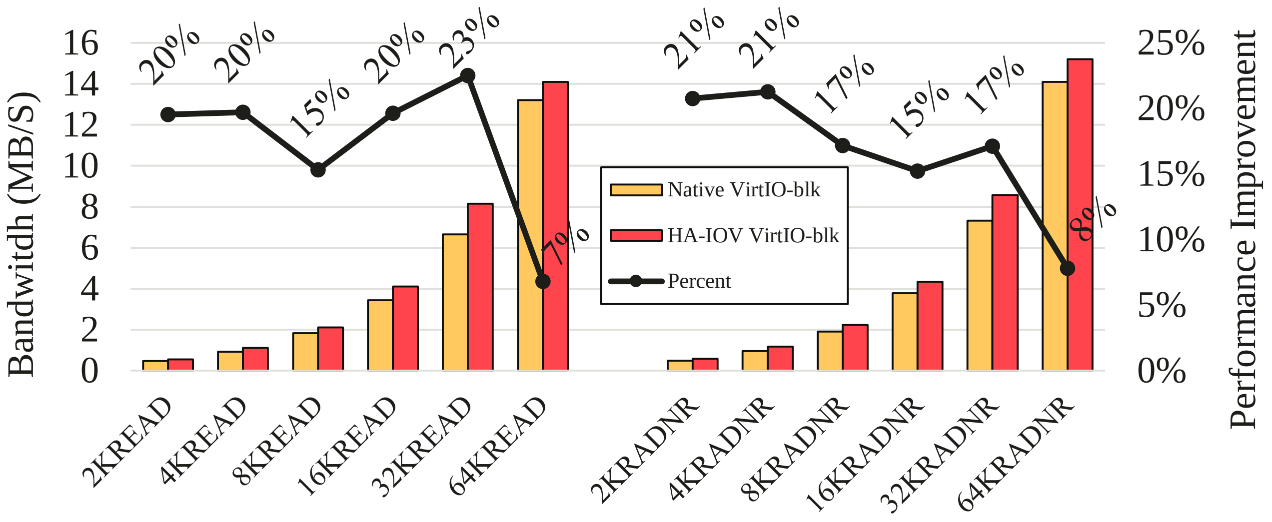


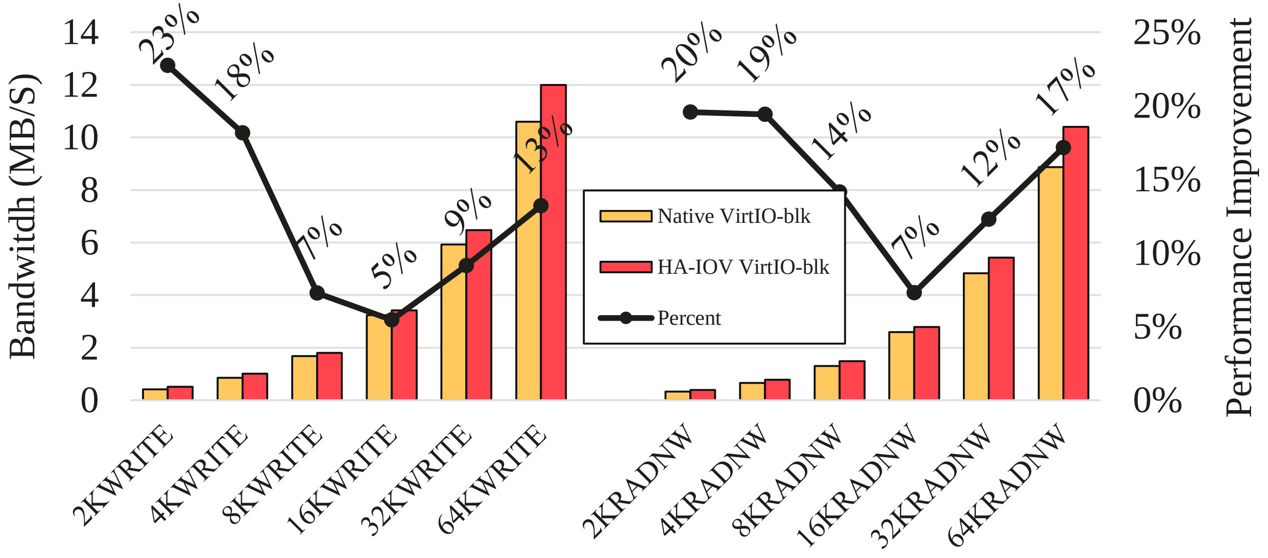
HA-IOV for Paravirtual I/O Devices in Userspace

- Evaluation
  - ➢ Environment
    1. Hisilicon Kunpeng 920 2600MHz
    2. RISC-V QEMU Emulator V0.5.1
    3. RISC-V KVMTOOL V1
    4. RISC-V KVM V10
    5. RISC-V HostOS with 4 CPU 2048M
    6. RISC-V GuestOS with 1 CPU 1024M
  - ➢ Experiment

    FIO configured with read, write, randread and randwrite
  - ➢ Results

    Performance of HA-IOV based virtio-blk increase 20% on average when message sizes are small (the higher the better)





Contents

Conclusion and Future Work

- Conclusion
  - Proposing hardware-assisted technique for I/O virtualization, including UER, VEI and HCS.
  - Enhancing performance of full emulated I/O devices and paravirtual I/O devices in both kernel and userspace.
  - Improving utilization of physical CPU resources by freeing up polling CPU cores.

Conclusion and Future Work

- Conclusion
  - Proposing hardware-assisted technique for I/O virtualization, including UER, VEI and HCS.
  - Enhancing performance of full emulated I/O devices and paravirtual I/O devices in both kernel and userspace.
  - Improving utilization of physical CPU resources by freeing up polling CPU cores.
- Future work
  - Optimization of VEI map query latency in hardware
  - Providing CPU affinity policies of VEI for balancing loads
  - Enhancing security of HA-IOV

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI