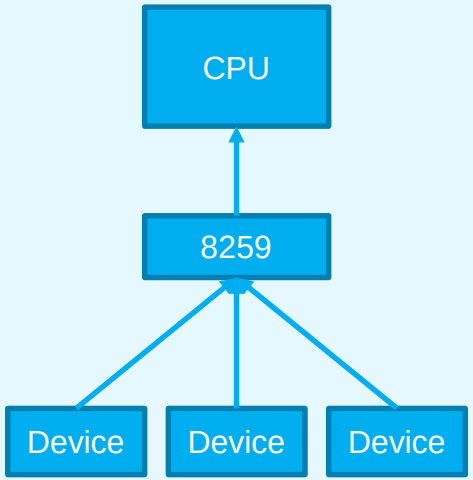
Device Interrupt Virtualization for Container/Serverless

Chao Peng(chao.p.peng@intel.com)

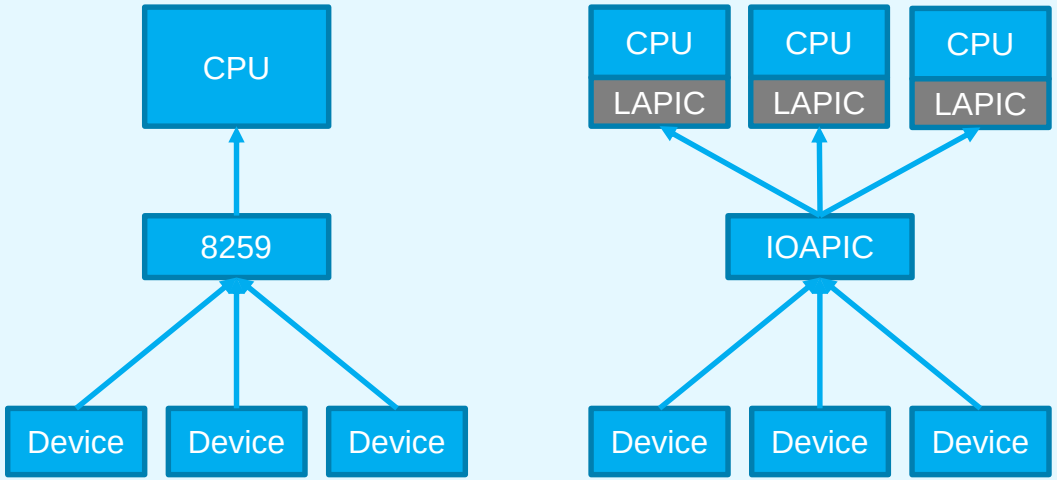
Jing Liu(jing2.liu@intel.com)

Interrupt(x86) Recap



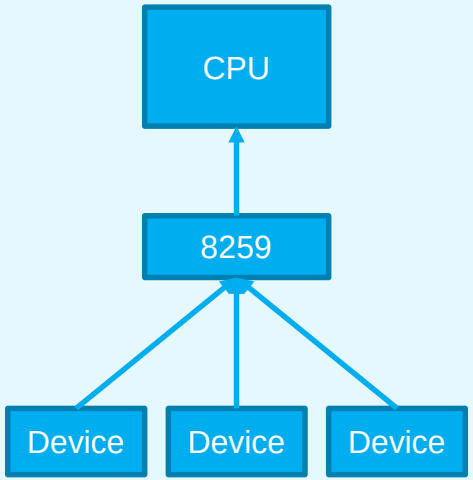
PIC

Interrupt(x86) Recap

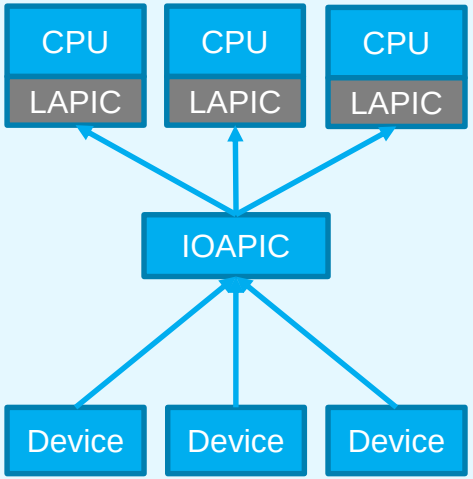


PIC **APIC**

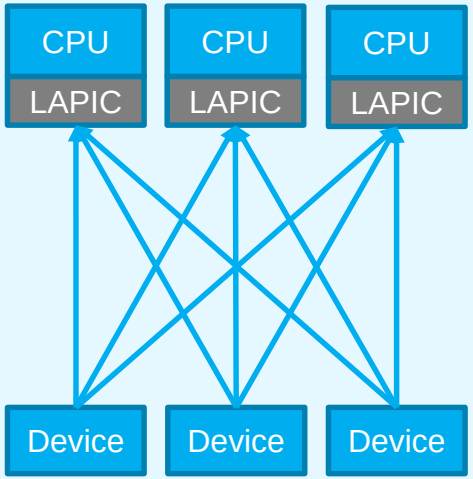
Interrupt(x86) Recap



PIC

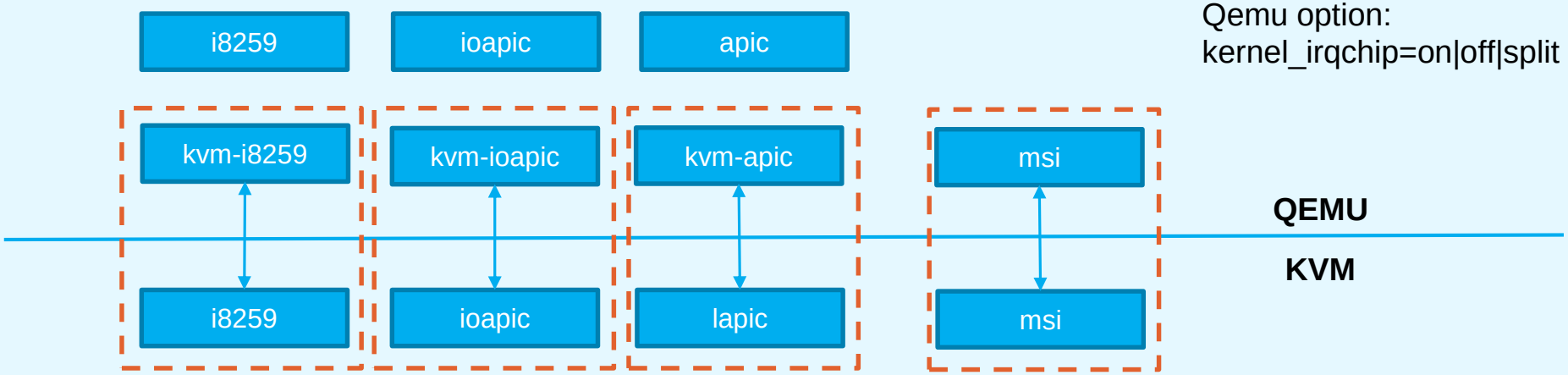


APIC



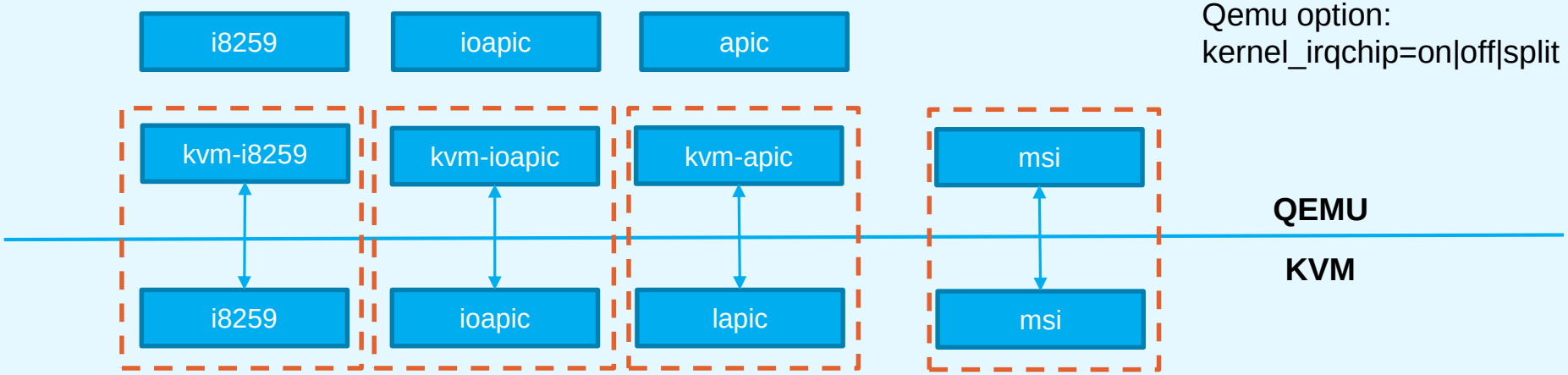
MSI

Interrupt Controllers in KVM/QEMU



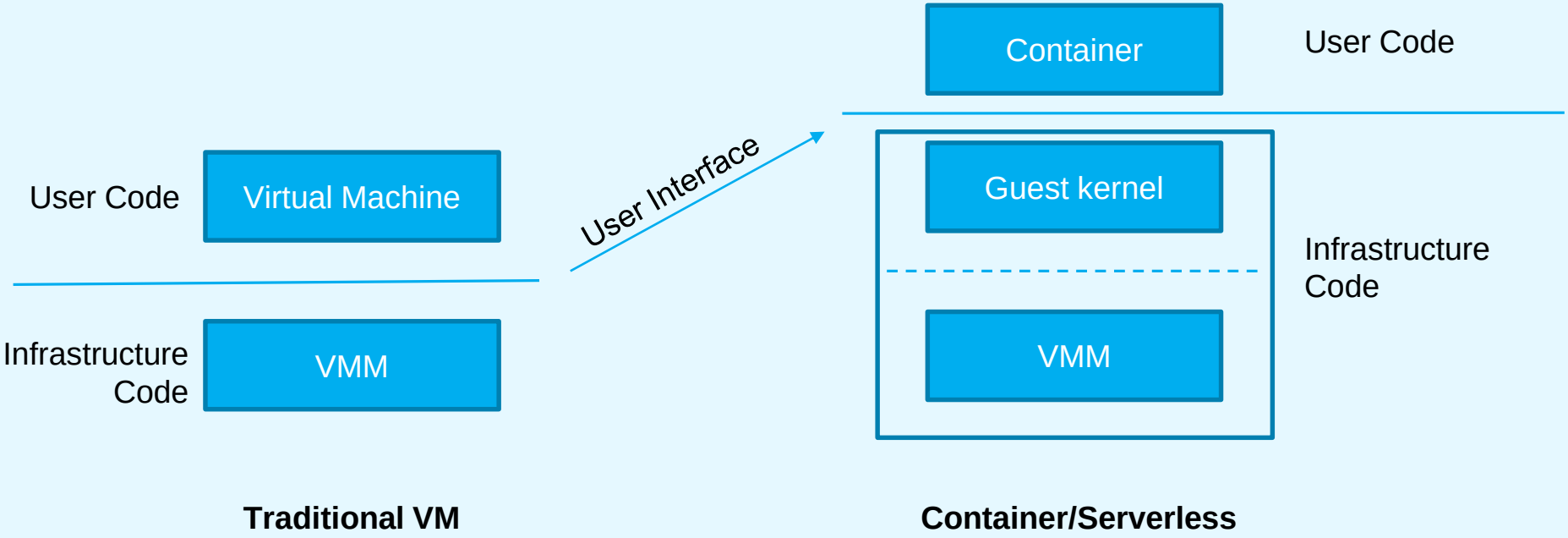
Qemu option:
kernel_irqchip=on|off|split

Interrupt Controllers in KVM/QEMU

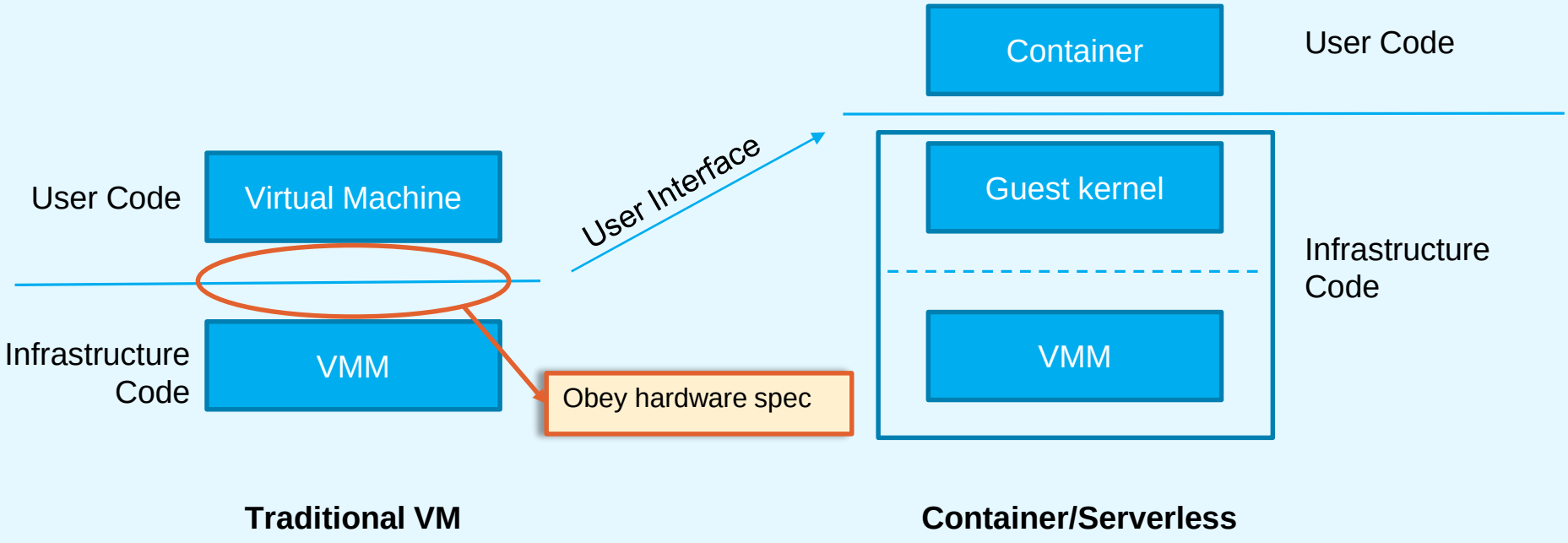


KVM/QEMU provide both legacy/modern Interrupt emulations for general purpose usages

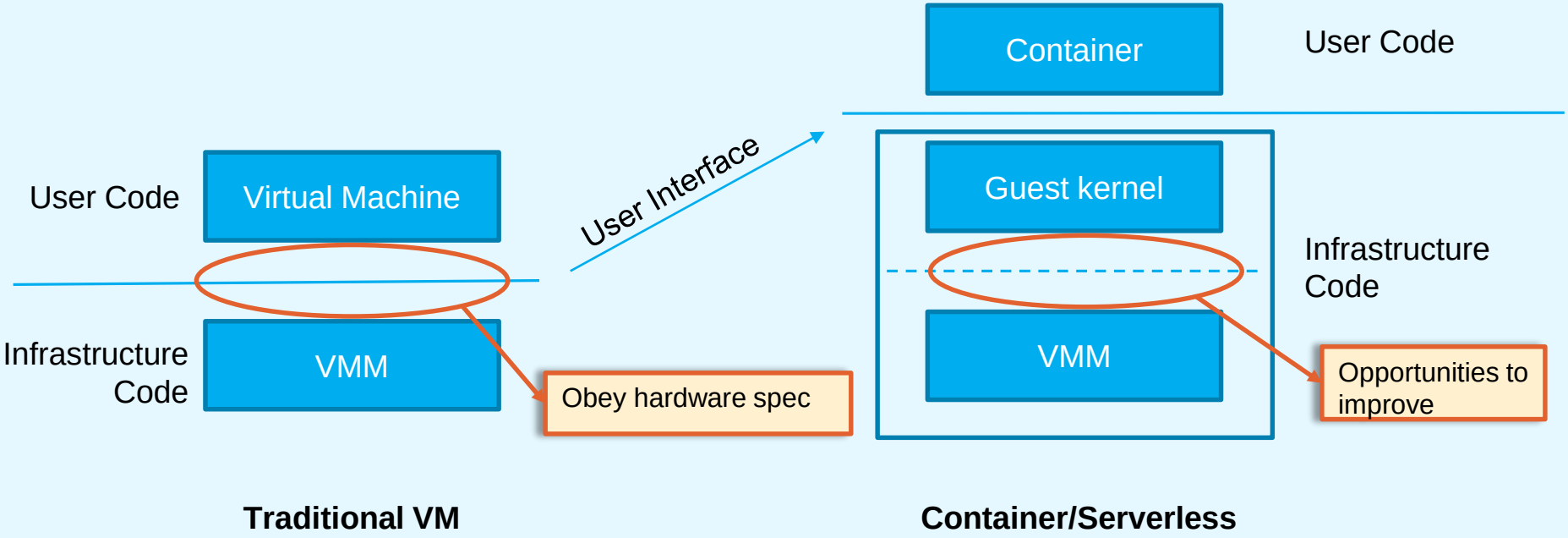
Our Focus: Container/Serverless



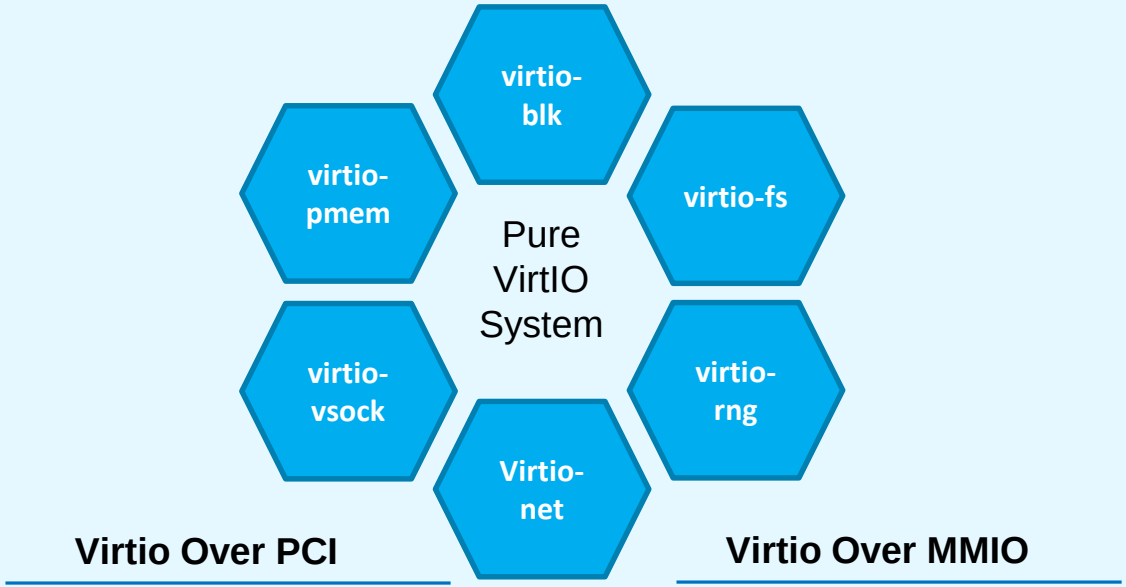
Our Focus: Container/Serverless



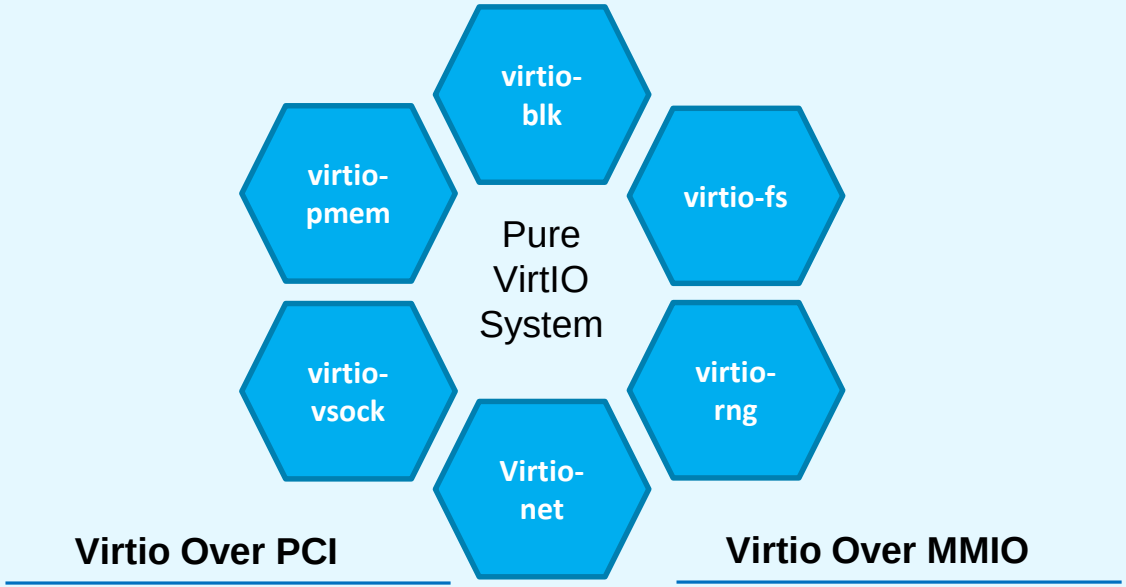
Our Focus: Container/Serverless



Our Focus: Limited Device Type



Our Focus: Limited Device Type



We have built a pure Virtio Over PCI system and continue to seek a better solution based on Virtio Over MMIO

Even Lighter: Virtio Over MMIO

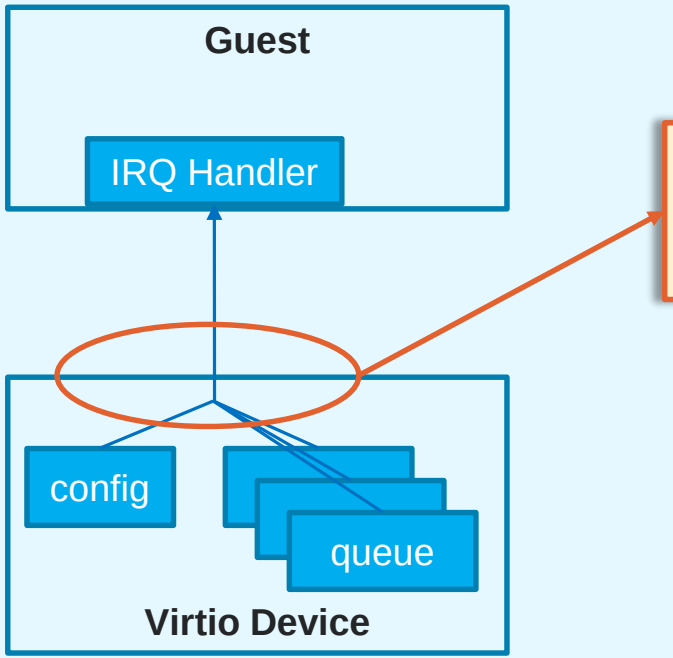
	Virtio Over PCI	Virtio Over MMIO
number of files(Linux)	161	1
line of code (Linux)	78237	538
number of files(QEMU)	24	1
line of code (QEMU)	8952	421

- Less code means
  - Shorter boot time
  - Higher density
  - Smaller attack surface

Minimal Requirement

- Multiple Interrupts
  - Device can have multiple interrupts
  - Performance
- Interrupt Balancing
  - Multi-vCPU is quite common
- Fast Interrupt Delivery
  - Can we make a single interrupt delivery faster?

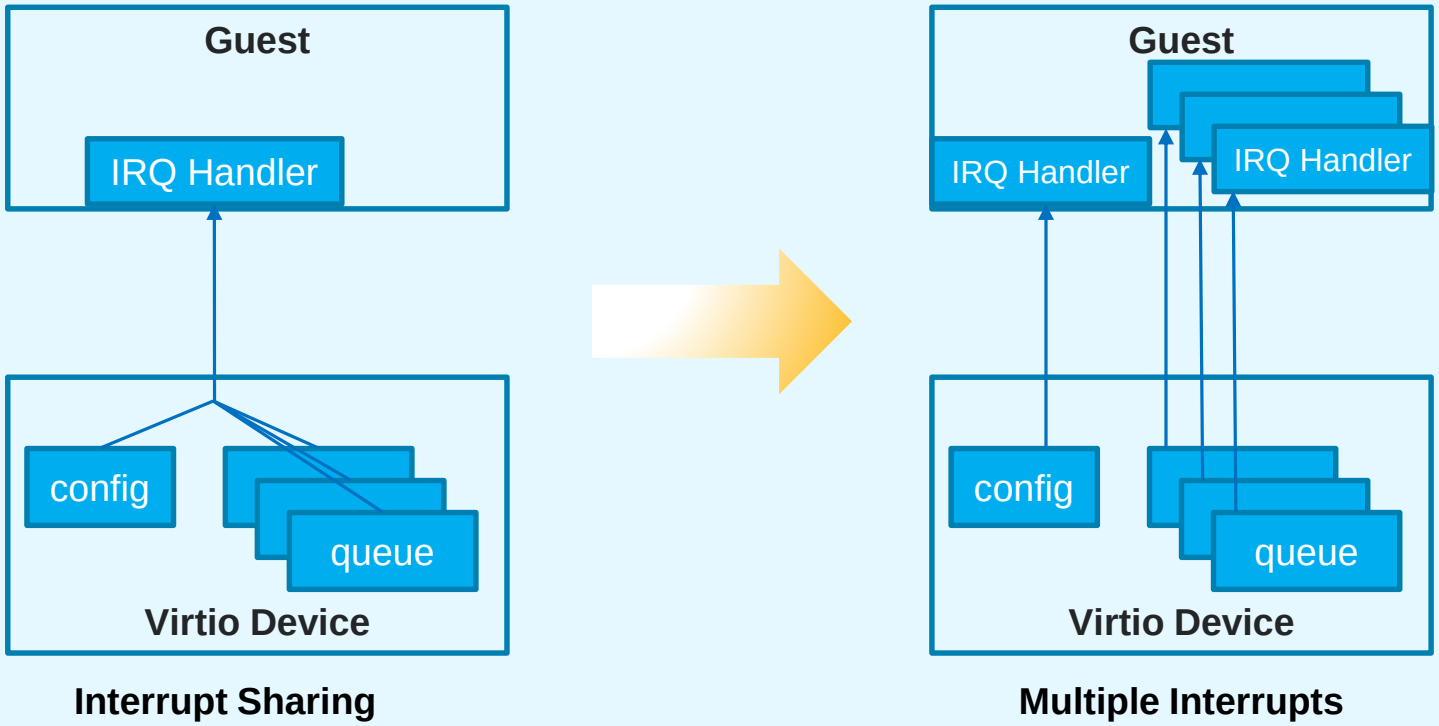
Virtio Over MMIO: Multiple Interrupts



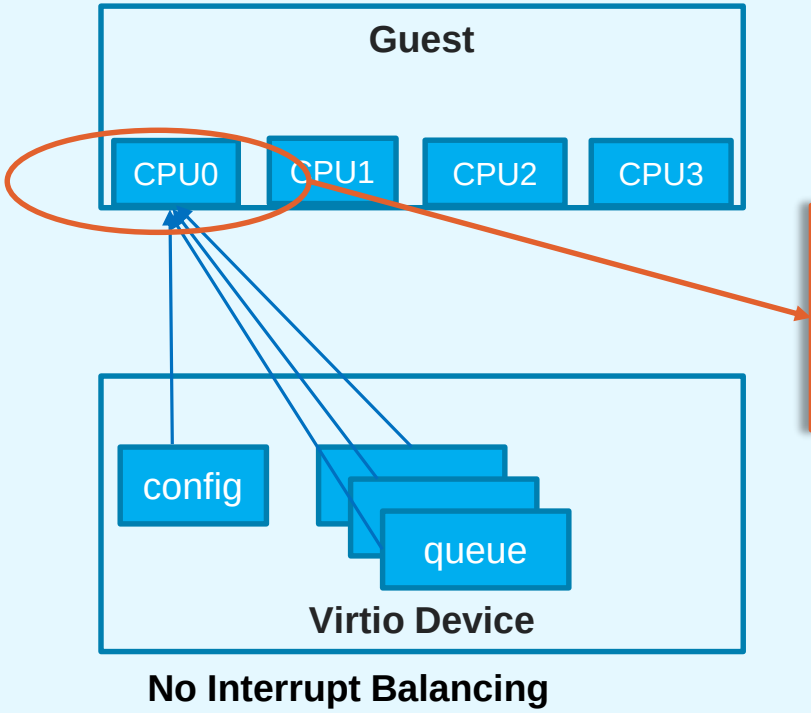
Sharing is good – except when it comes to interrupt.

Current Virtio Over MMIO

Virtio Over MMIO: Multiple Interrupts



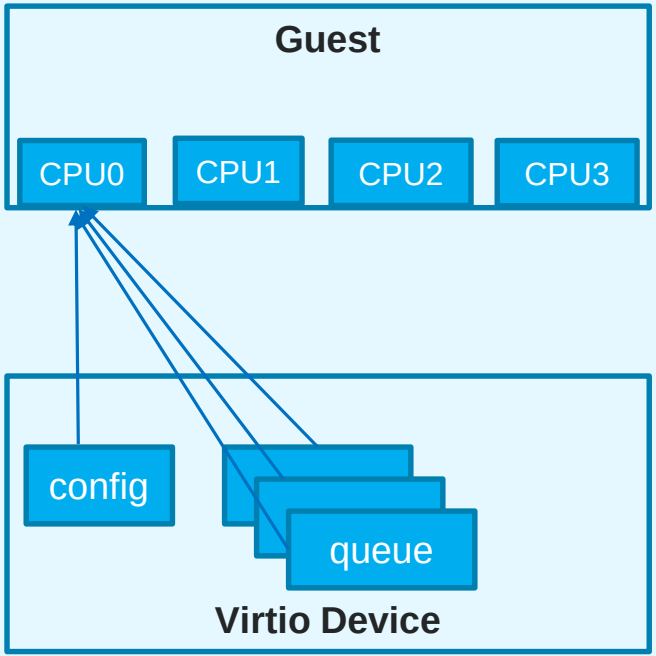
Virtio Over MMIO: Interrupt Balancing



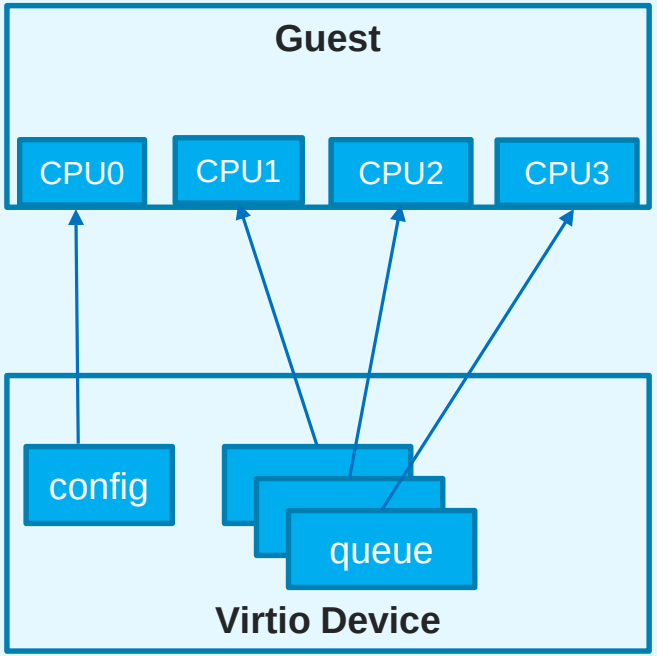
No Interrupt Balancing

All Interrupts are delivered to the same vCPU and there is no way for Guest to set interrupt affinity.

Virtio Over MMIO: Interrupt Balancing



No Interrupt Balancing



With Interrupt Balancing

Solution 1: Virtio MMIO + MSI

- Multiple interrupts
  - A new feature bit for capability discovery
  - config_msix_vector for configuration
  - queue_msix_vector for each queue
- Interrupt balancing
  - Encoded in MSI message address/data
- Interrupt status/control:
  - enable/mask/pending bits

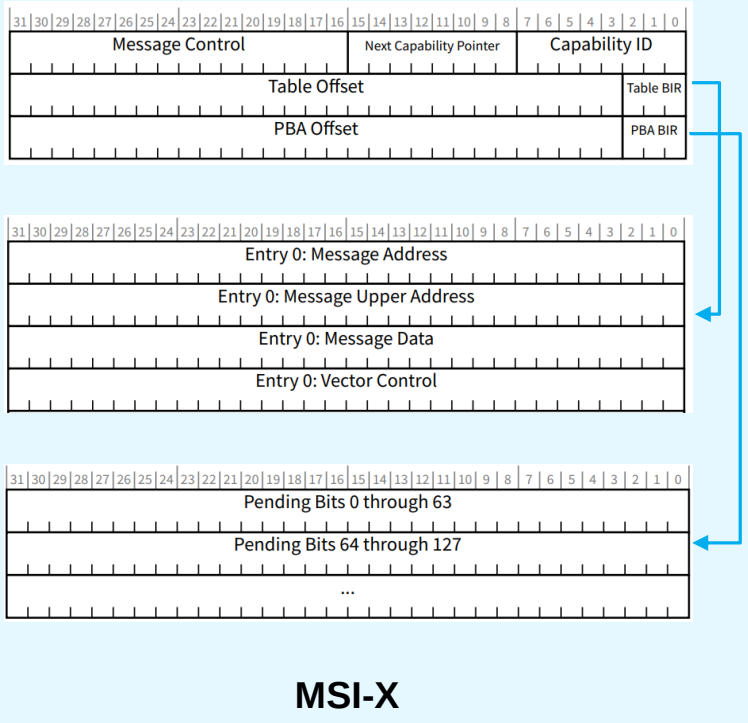
Good sharing with existing PCI code across QEMU/KVM/Guest kernel

Solution2: Virtio Interrupt Storage(VIS)

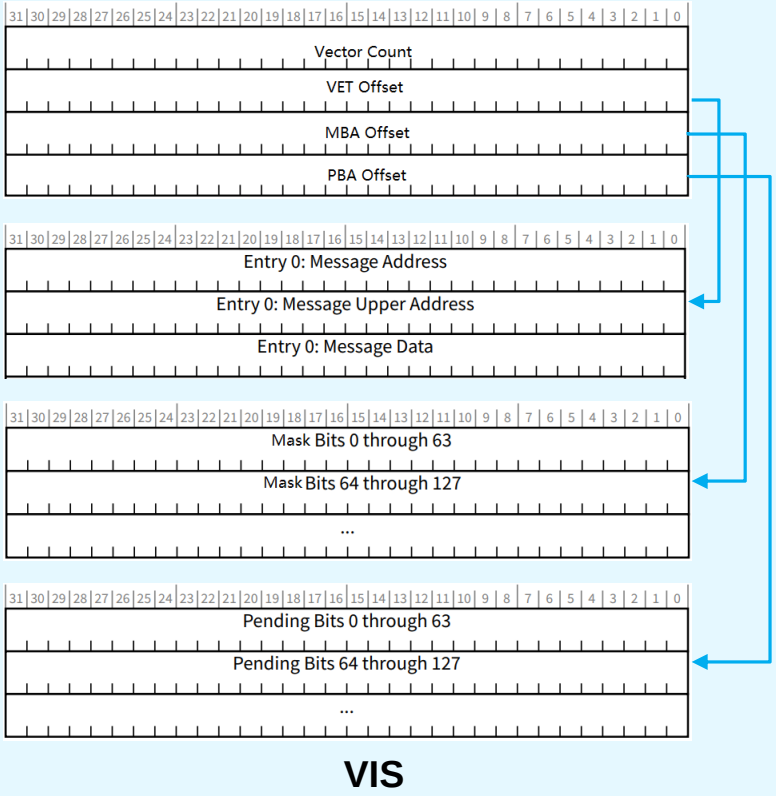
- A virtio native Interrupt mechanism
  - Mainly for virtio MMIO, may be extended to other transports
- Reuse a lot from MSI
  - Still some code sharing
- Improvement over MSI
  - No PCI concepts like MSI capability
  - Simplify/re-organize register layout to reduce exits
    - E.g. separate mask bit from MSI-X table entry

Virtio native and virtualization-optimized interrupt mechanism

Virtio Interrupt Storage(VIS)

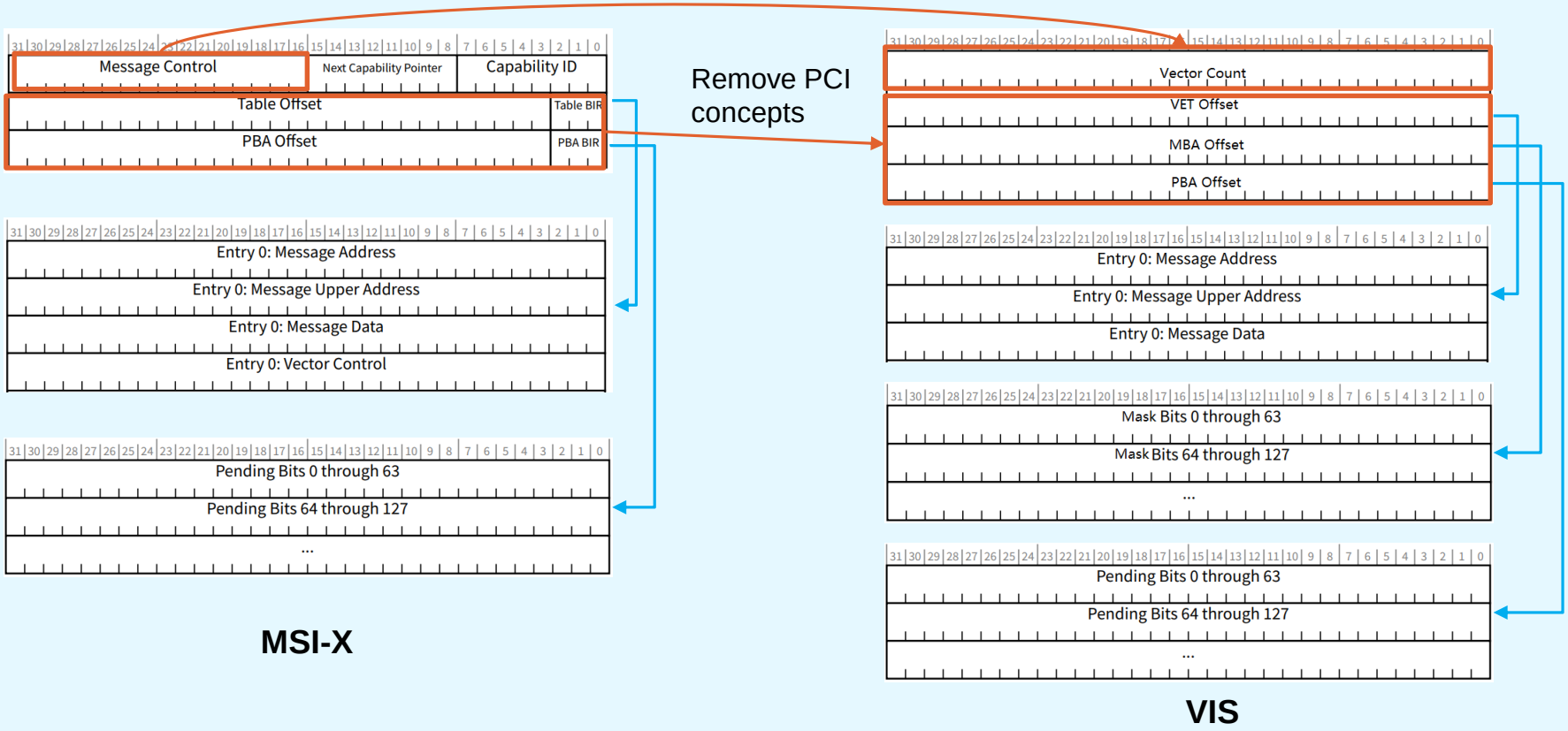


MSI-X

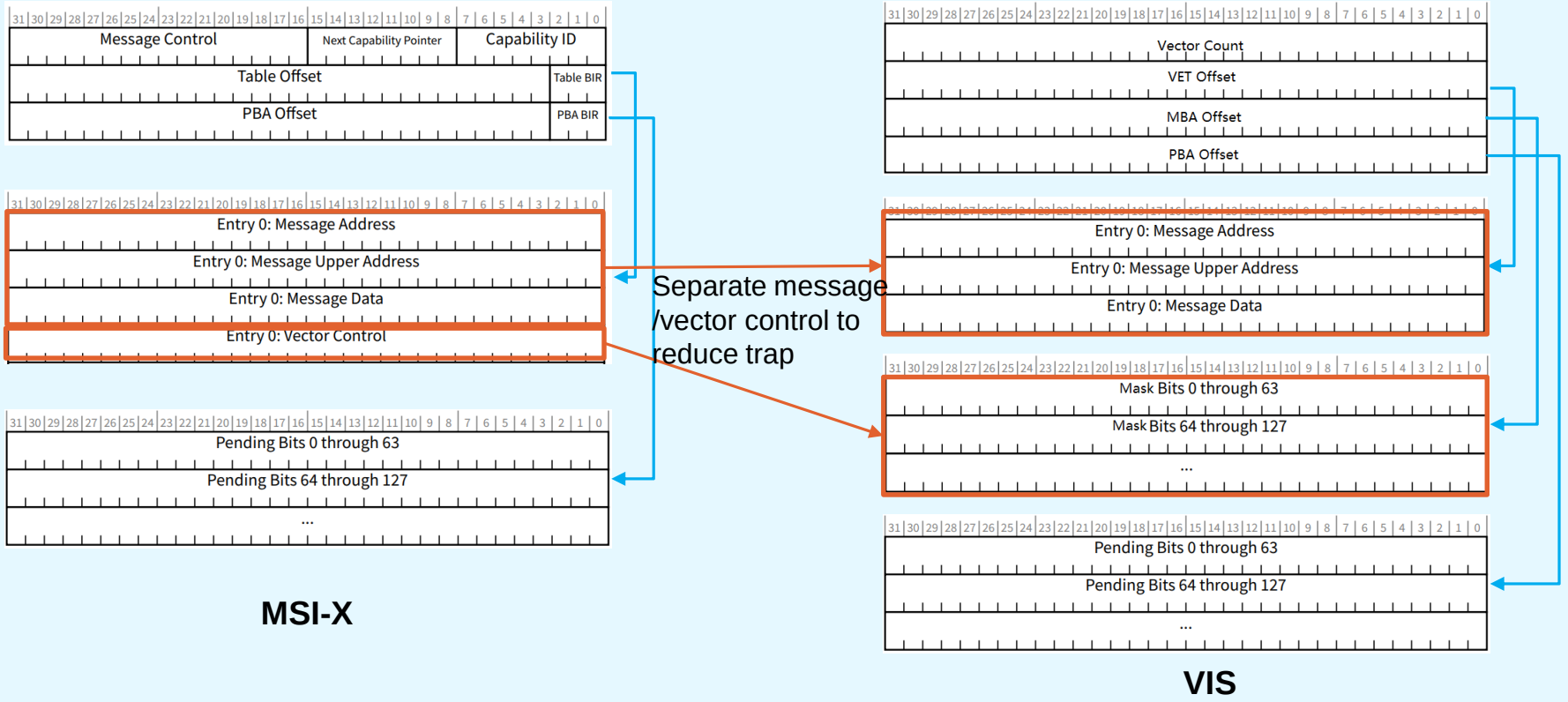


VIS

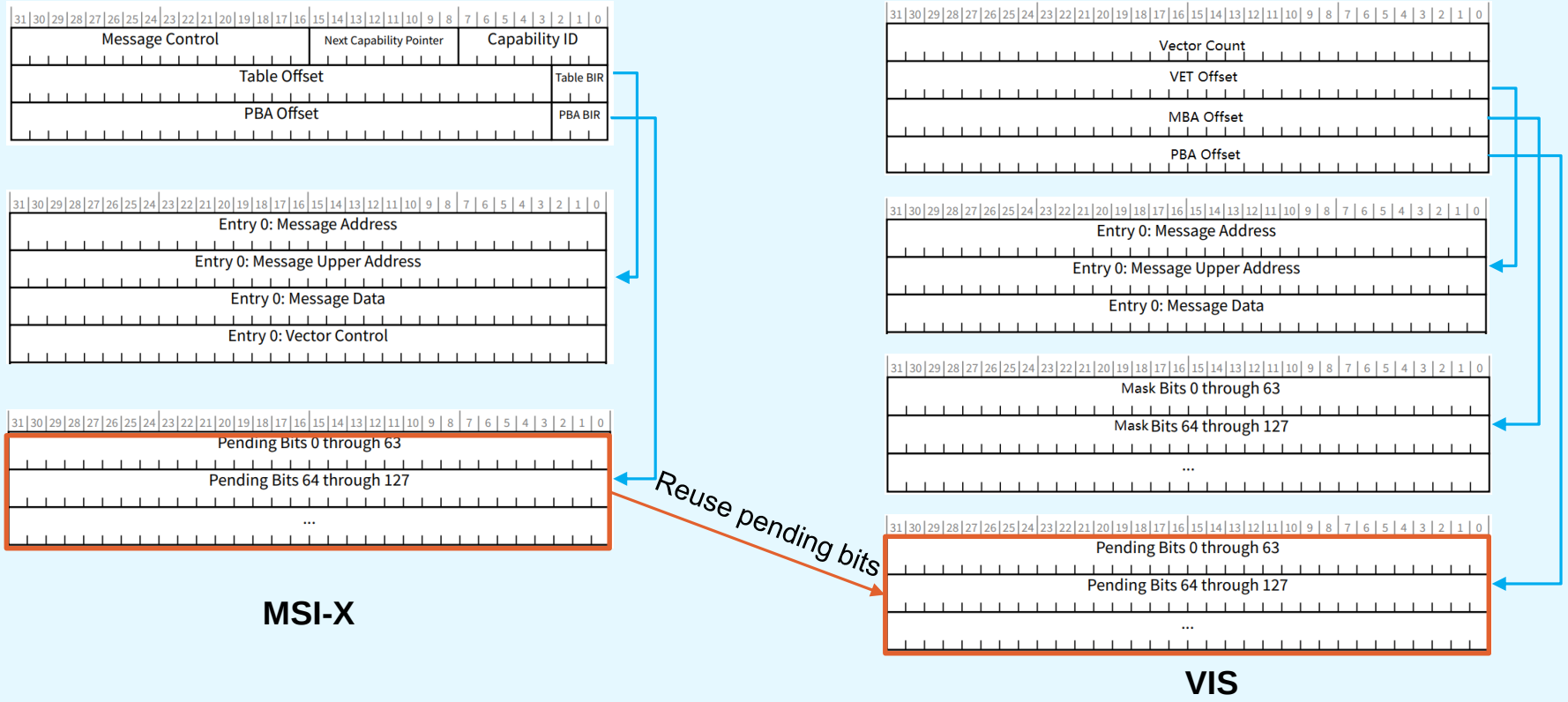
Virtio Interrupt Storage(VIS)



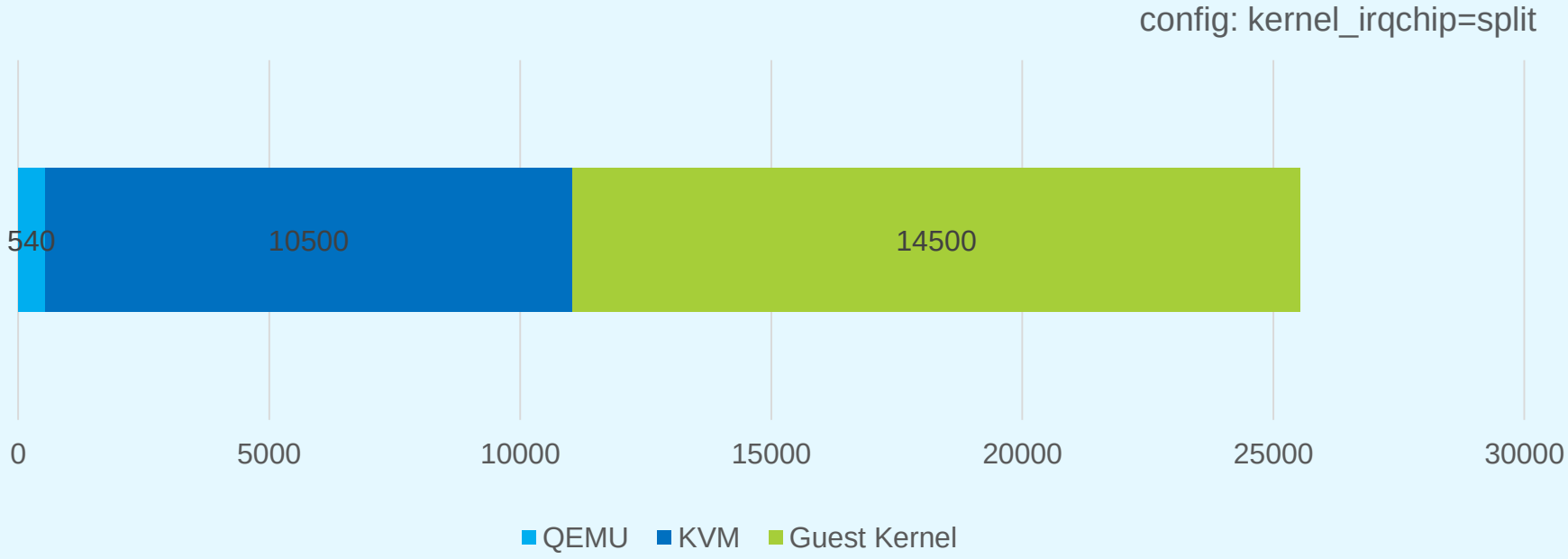
Virtio Interrupt Storage(VIS)



Virtio Interrupt Storage(VIS)

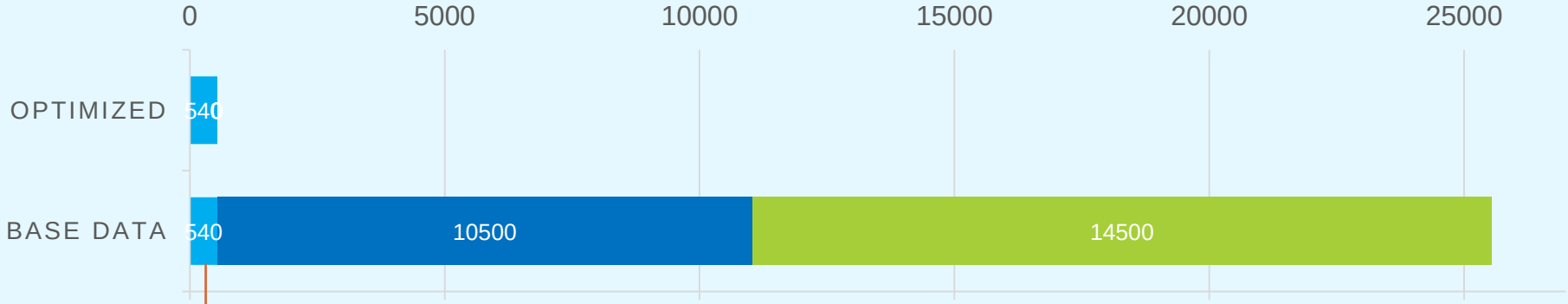


Virtio Over MMIO: Interrupt Delivery



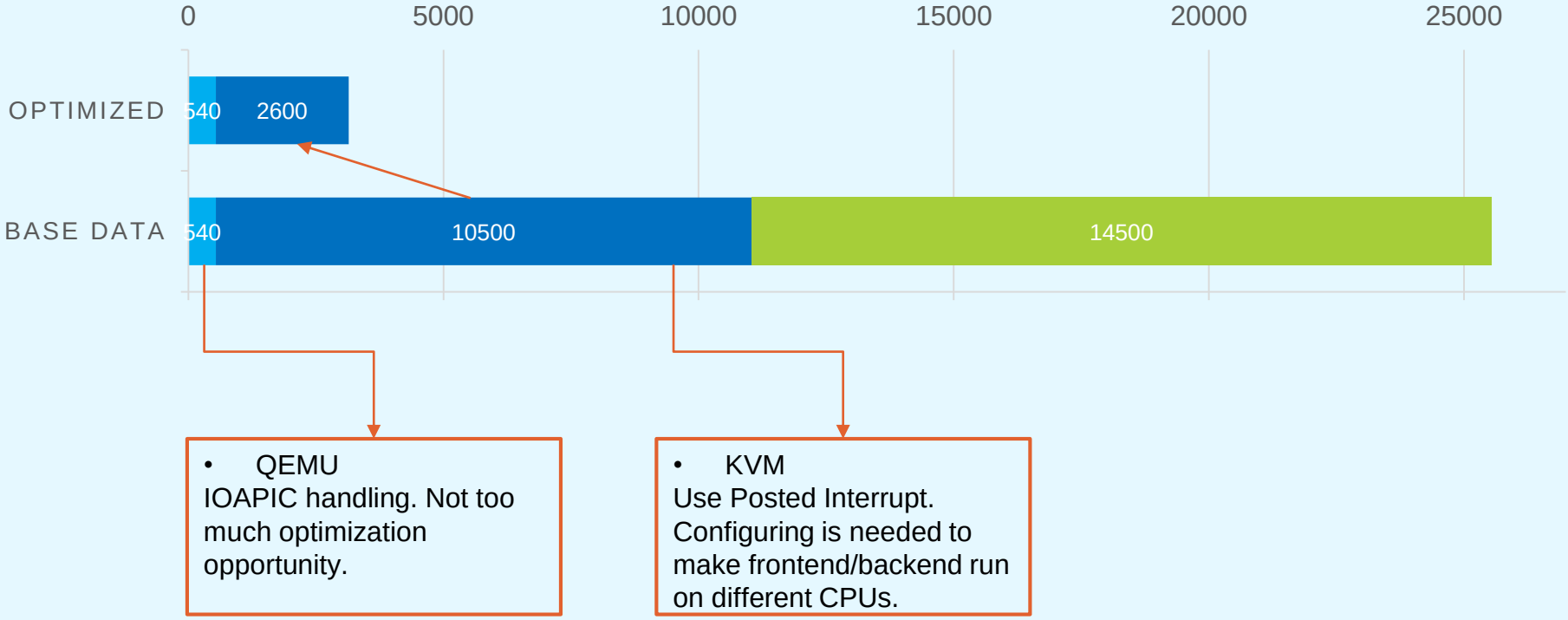
Cycles of delivering an interrupt

Interrupt Delivery Optimization

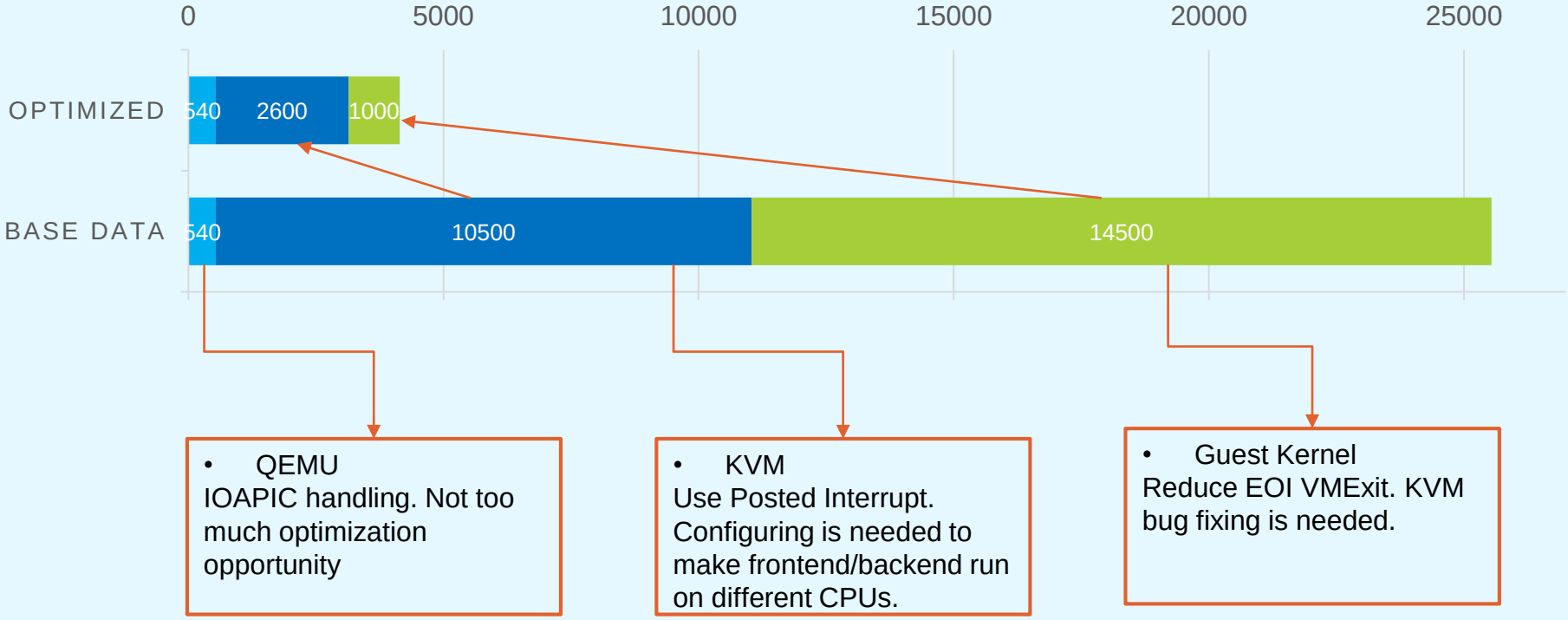


- QEMU IOAPIC handling. Not too much optimization opportunity.

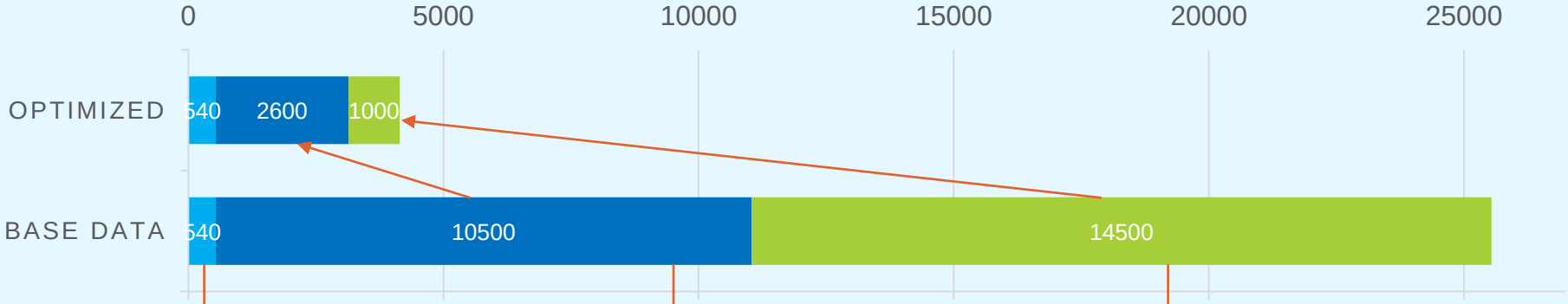
Interrupt Delivery Optimization



Interrupt Delivery Optimization



Interrupt Delivery Optimization



- QEMU
  IOAPIC handling. Not too much optimization opportunity.

- KVM
  Use Posted Interrupt. Configuring is needed to make frontend/backend run on different CPUs.

- Guest Kernel
  Reduce EOI VMExit. KVM bug fixing is needed.

Totally reduced: 21400 cycles = saved interrupt delivering time: 9ms

Summary

- We are building a new Interrupt system for Container / Serverless, with minimal features
  - Multiple interrupts
  - Interrupt balancing
- We prototyped several options
  - Pure virtio system with PCI and MMIO transport
  - virtio-mmio with MSI and a new virtio native interrupt
- We use hardware feature to improve interrupt delivering
  - Posted interrupt
  - EOI virtualization

What's next?

- It's the time to change the spec
  - Either the MSI or VIS can not work without spec change
  - Your opinions
- Upstream
  - We really want to contribute KVM/QEMU/kernel changes if people are interested
- Assigned device
  - Currently assigned devices is built on top of PCI
  - Can we do that with virtio-mmio?

Legal Disclaimer

- Intel provides these materials as-is, with no express or implied warranties.
- All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://intel.com.
- Some results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.
- Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
- © Intel Corporation