Distributed QEMU

Yubin Chen <binsschen@sjtu.edu.cn>

Zhuocheng Ding <tcbbd@sjtu.edu.cn>

Tcloud Lab, Shanghai Jiao Tong University

Agenda

- **Demo**
- Motivation & Challenge
- Design Detail
  - CPU
  - I/O
  - Memory
- Performance

Distributed QEMU Demo

Jin Zhang<jzhang3002@sjtu.edu.cn>

Agenda

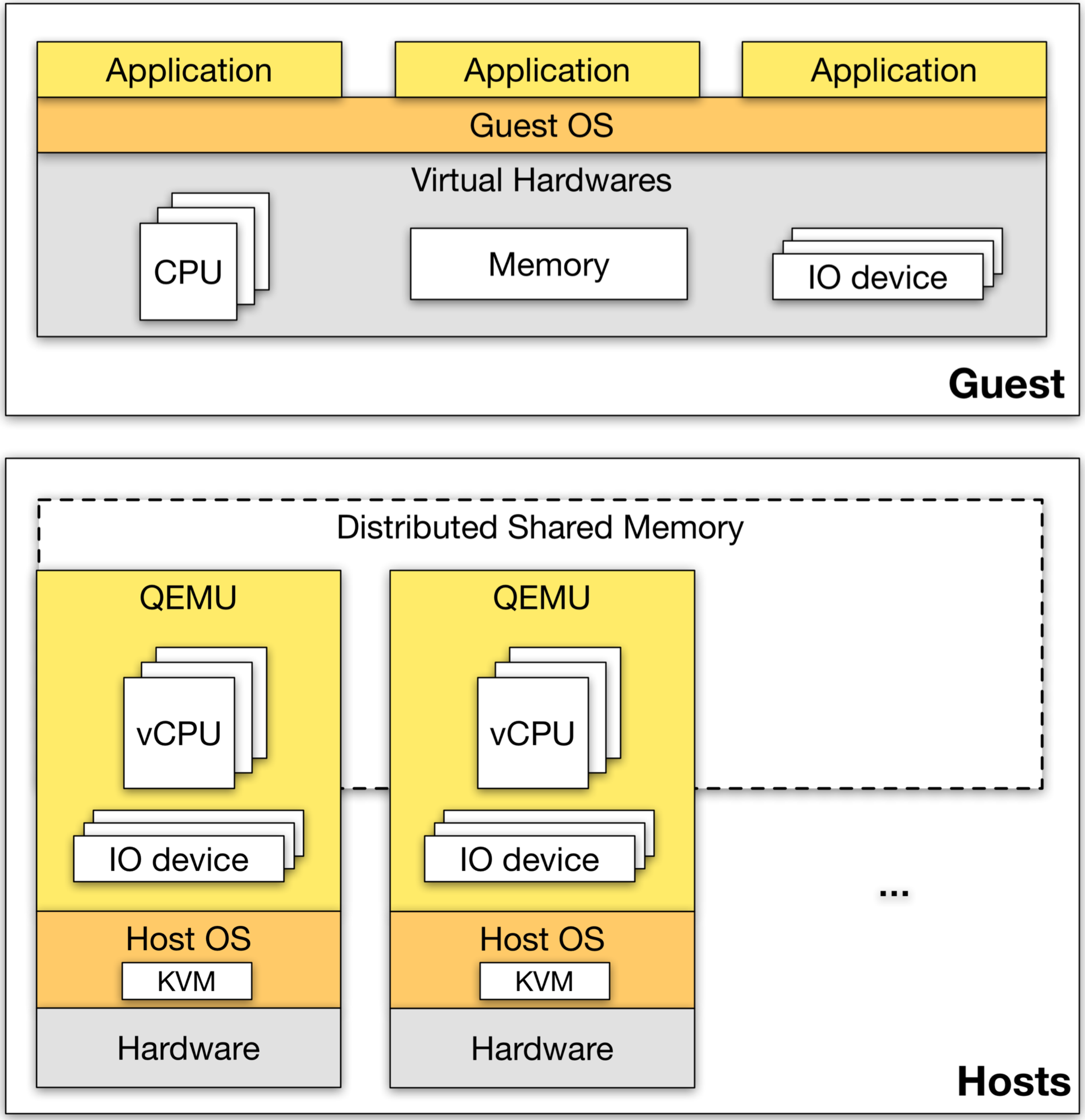
- Demo
- **Motivation & Challenge**
- Design Detail
  - CPU
  - I/O
  - Memory
- Performance

Motivation

- Deal with the big data problem
  - Scale-up: Mainframes
    - Low cost-performance ratios
    - Physical limitation
  - Scale-out: Data processing frameworks, e.g. MapReduce, Spark
    - Programming Model Complexity
    - Runtime Overhead
    - Cannot be rewritten

Motivation

- Distributed QEMU
  - Run across multiple physical machines
  - Aggregate **CPU** / **Memory** / **IO devices** resources from different machines



Challenges

- Communication among different QEMUs
  - vCPUs
  - vCPUs and I/O devices
- Synchronization among different QEMUs
  - Memory
  - Time

Agenda

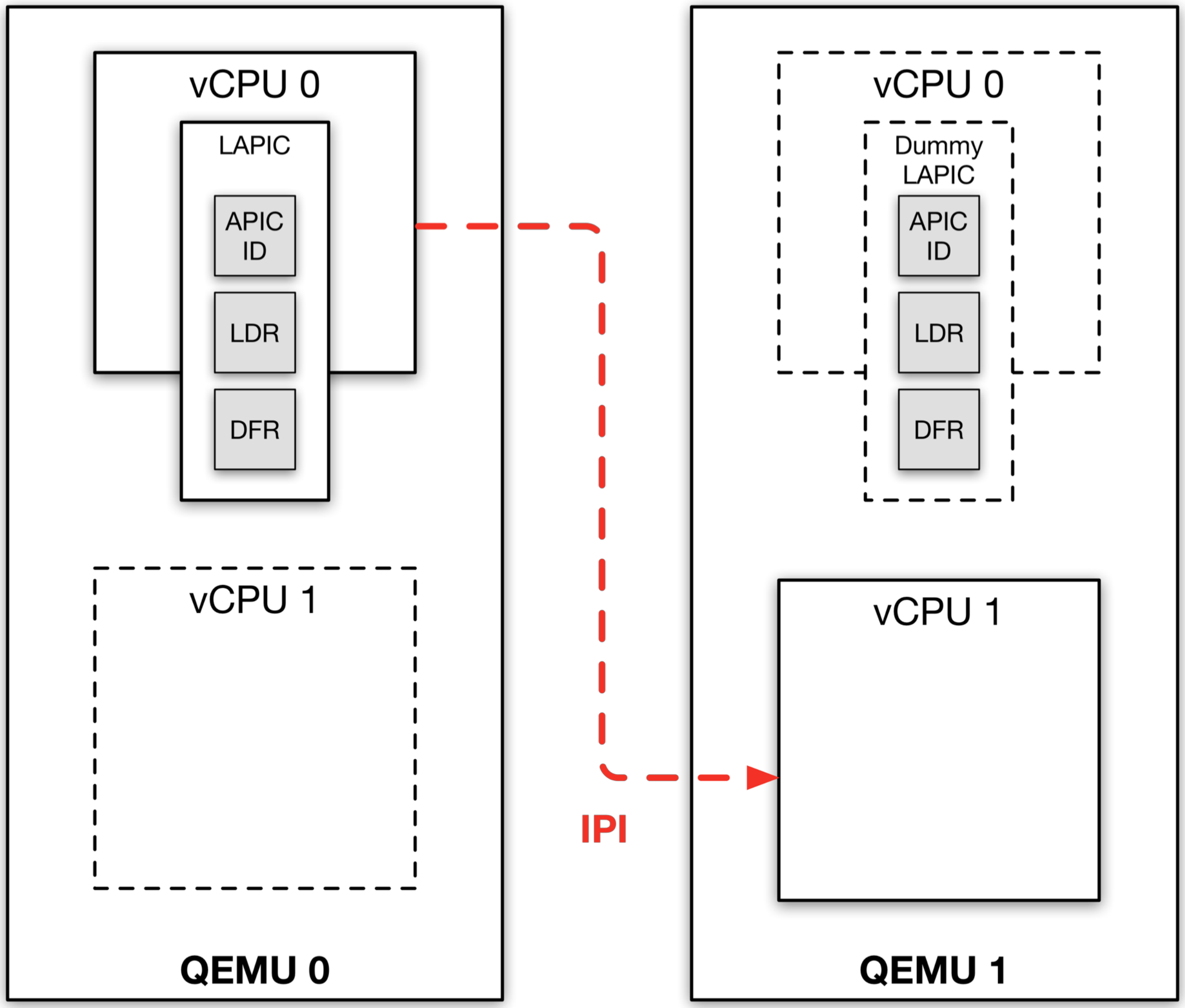
- Demo
- Motivation & Challenge
- **Design Detail**
  - CPU
  - I/O
  - Memory
- Performance

Design: CPU

- Distributed vCPU
  - Each QEMU create full amounts of vCPU
  - But only some are **LOCAL**, while others are **REMOTE**
    - LOCAL
      - Run like original vCPUs
    - REMOTE
      - Never run, never dive into KVM

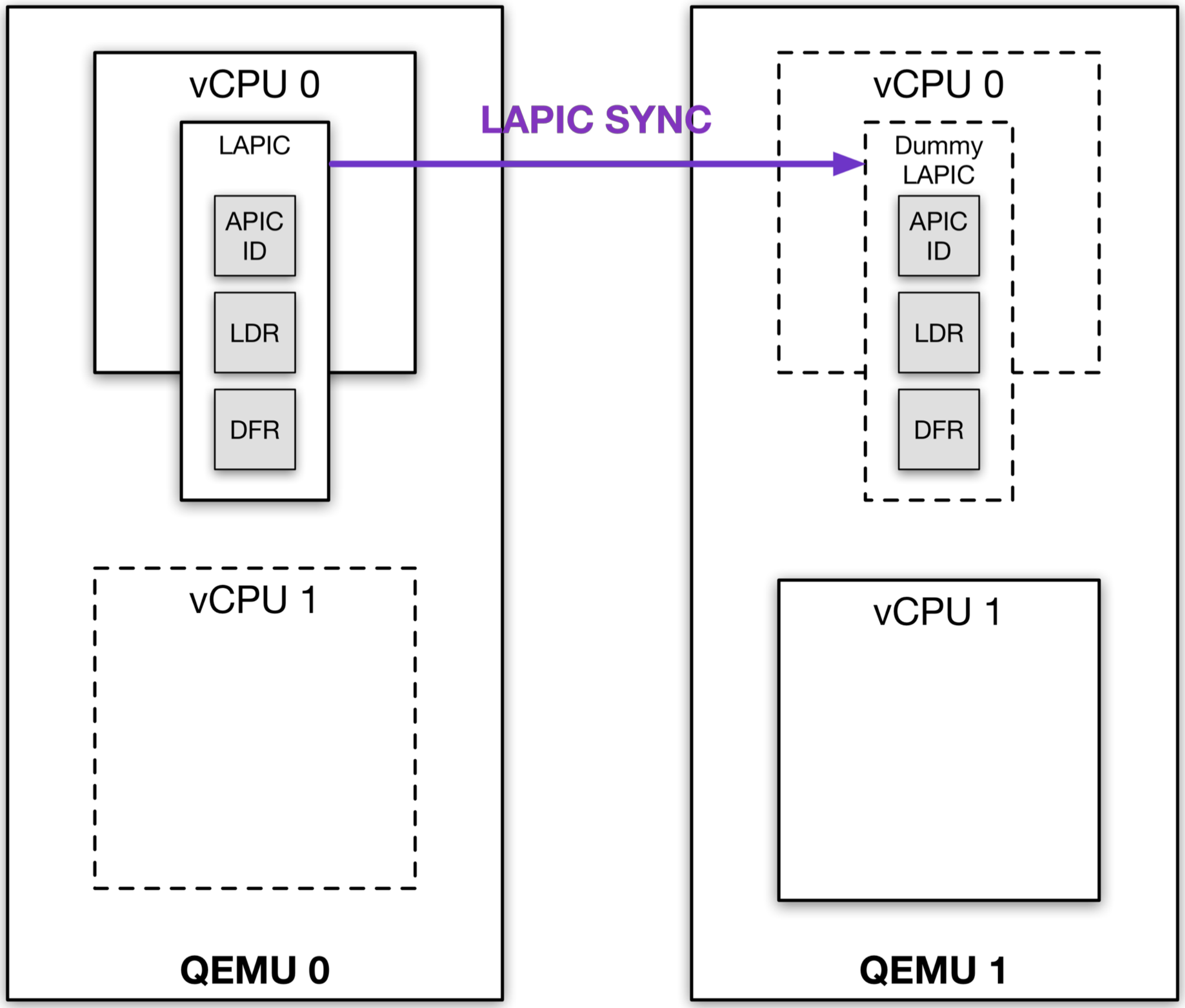
Design: CPU

- Inter Processor Interrupt (IPI) Forwarding
  - IPI to local vCPU
  - IPI to remote vCPU



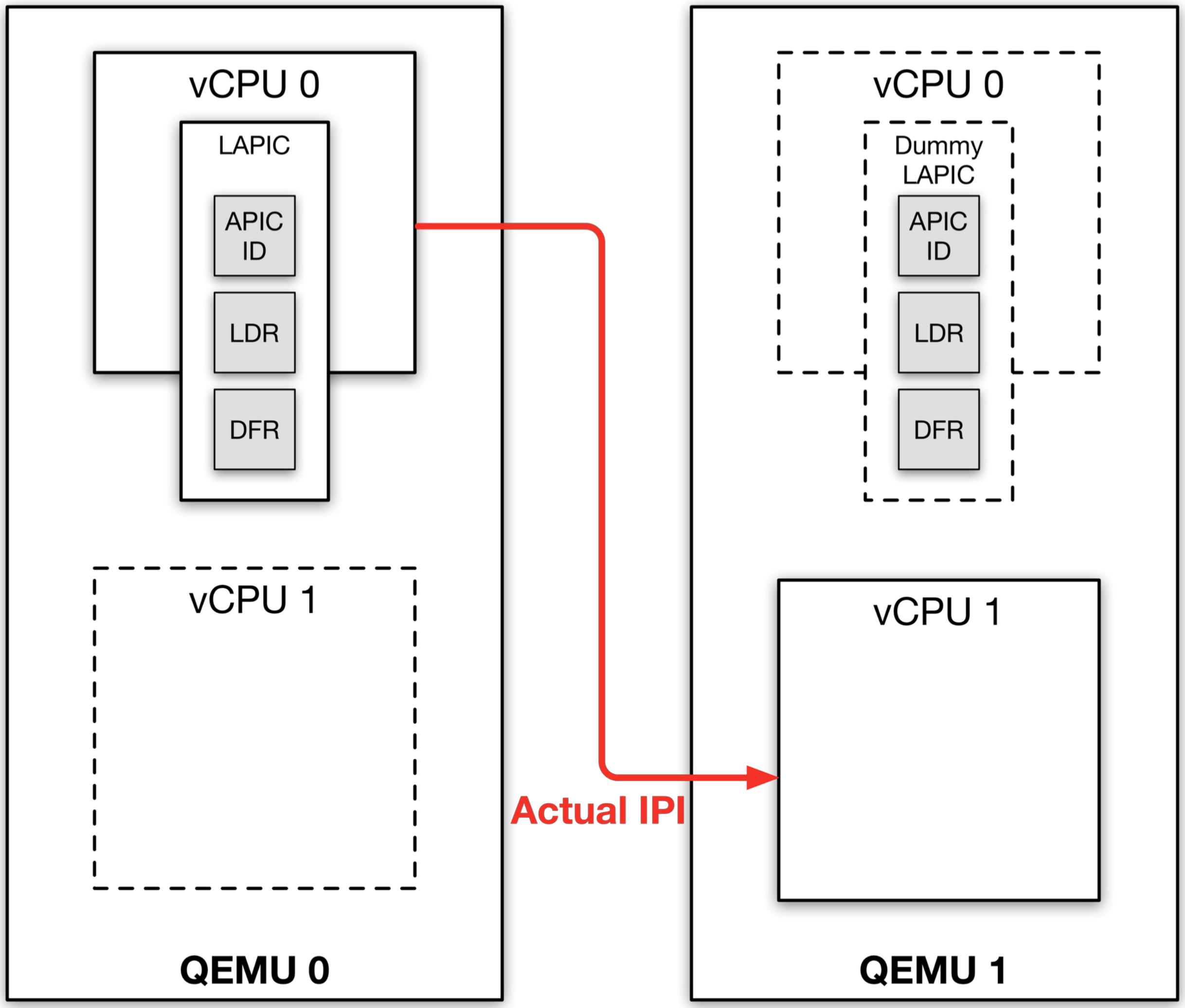
Design: CPU

- Inter Processor Interrupt (IPI) Forwarding
  - IPI to local vCPU
  - IPI to remote vCPU
  - Dummy APIC
    - Broadcast necessary registers to make all dummy APICs up to date



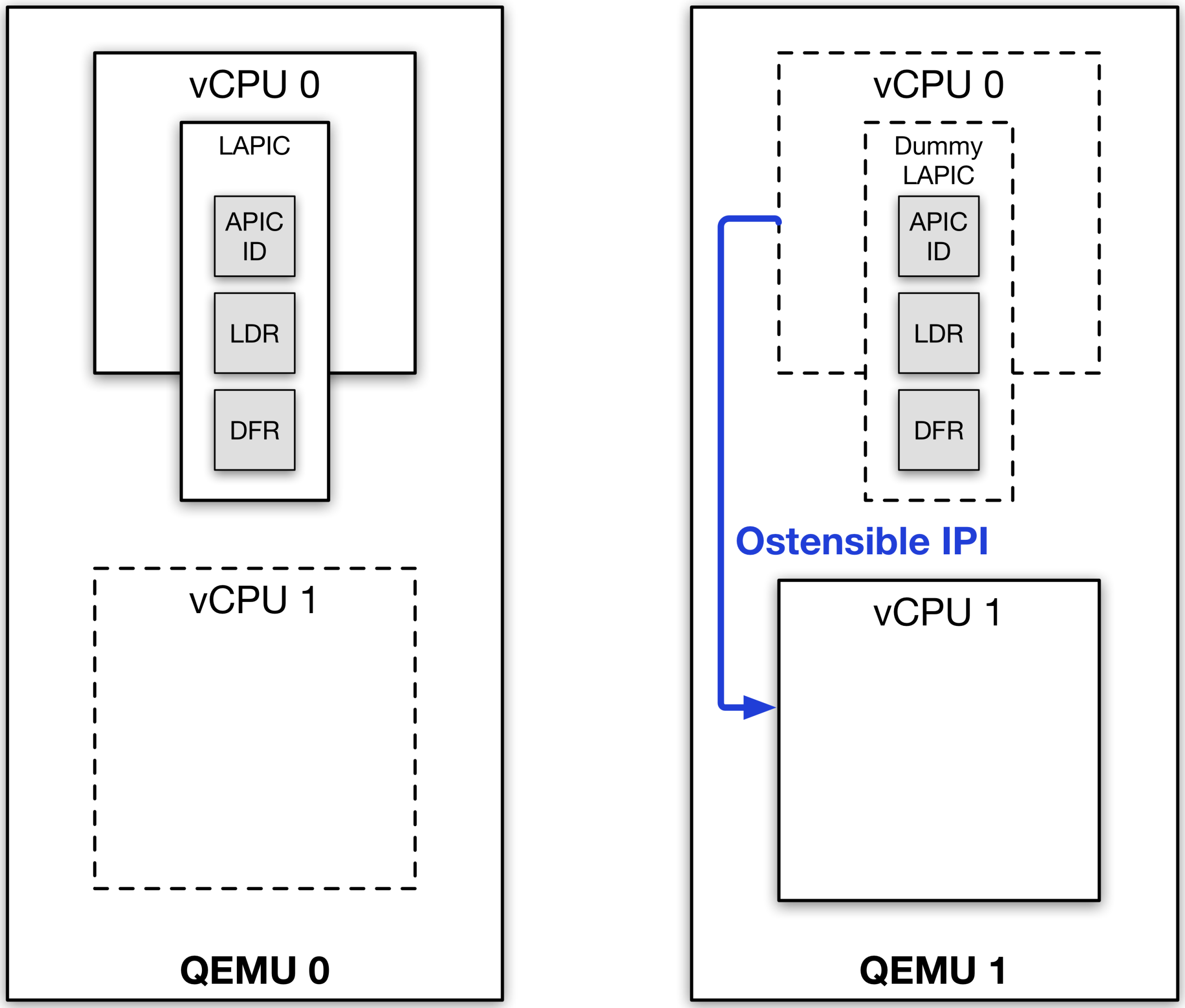
Design: CPU

- Inter Processor Interrupt (IPI) Forwarding
  - IPI to local vCPU
  - IPI to remote vCPU
  - Dummy APIC
    - Broadcast necessary registers to make all dummy APICs up to date



Design: CPU

- Inter Processor Interrupt (IPI) Forwarding
  - IPI to local vCPU
  - IPI to remote vCPU
  - Dummy APIC
    - Broadcast necessary registers to make all dummy APICs up to date

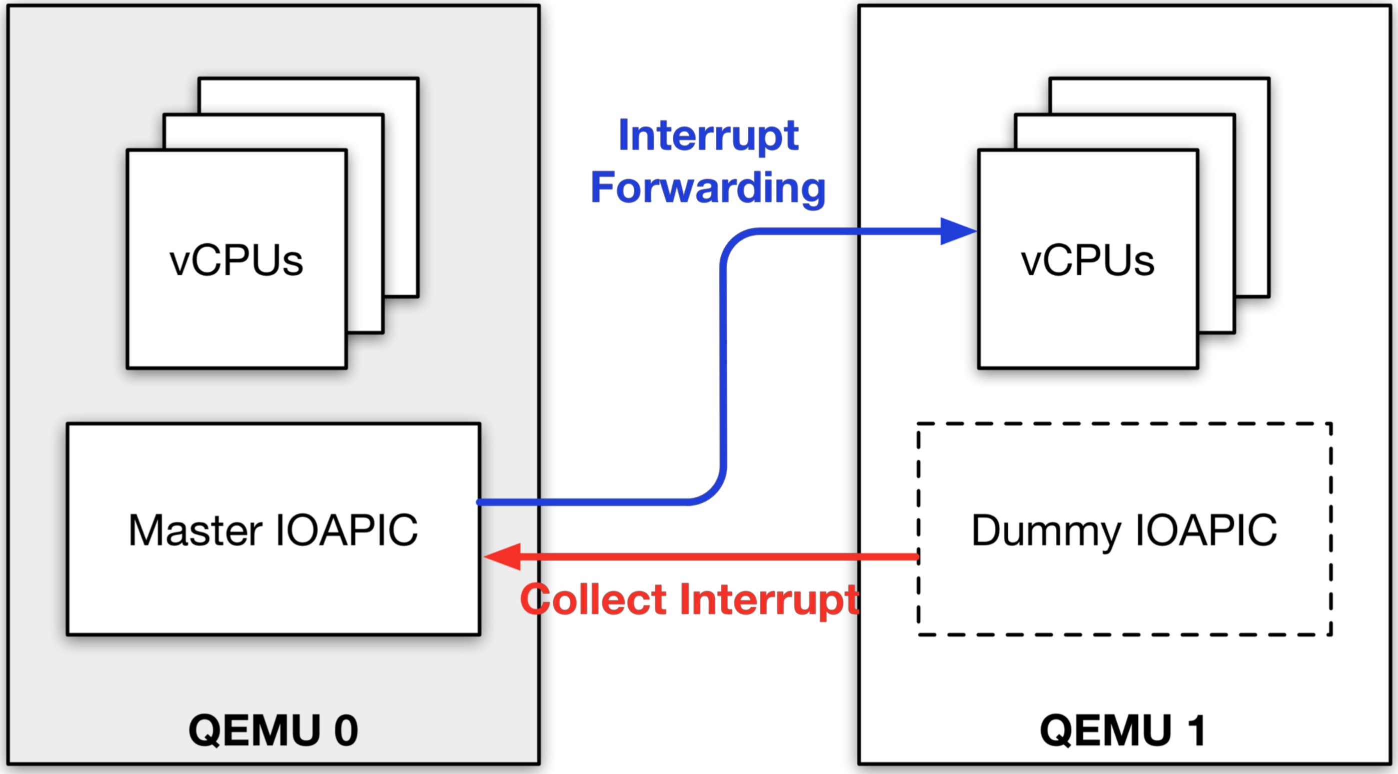


Design: CPU

- Interrupt Forwarding
  - IOAPIC based interrupts
    - Master IOAPIC

      Handle signals
    - Dummy IOAPIC

      Collect and forward them to the master IOAPIC
  - MSI



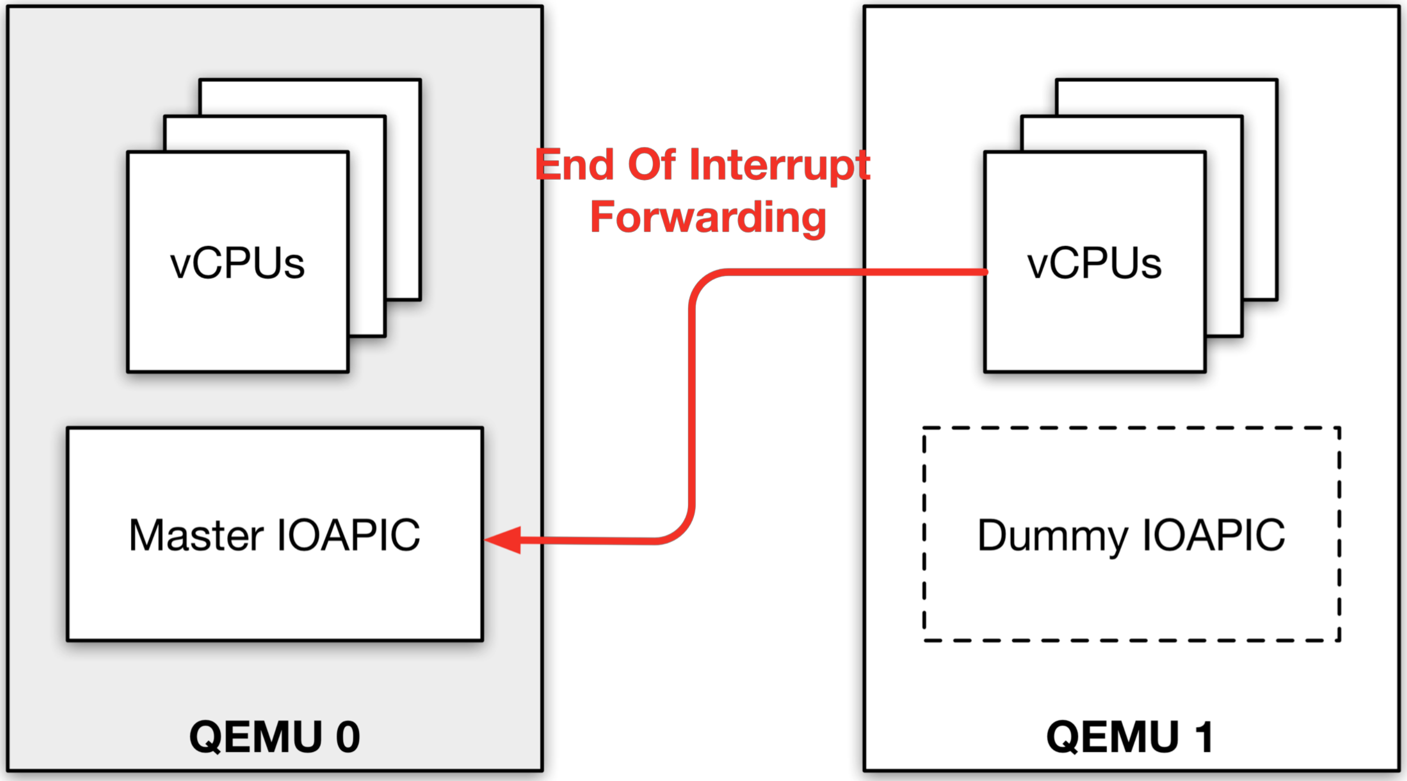
Design: CPU

- Interrupt Forwarding
  - IOAPIC based interrupts
    - Master IOAPIC

      Handle signals
    - Dummy IOAPIC

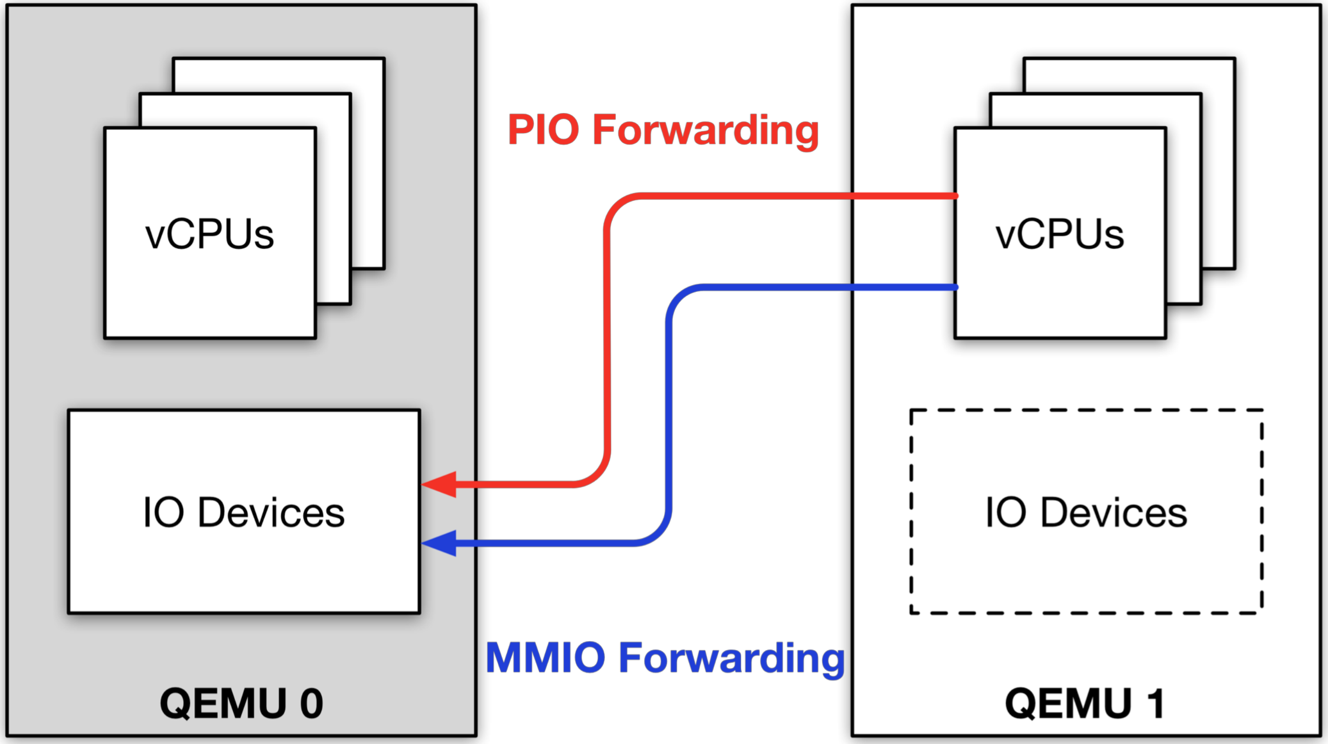
      Collect and forward them

      to the master IOAPIC
  - MSI



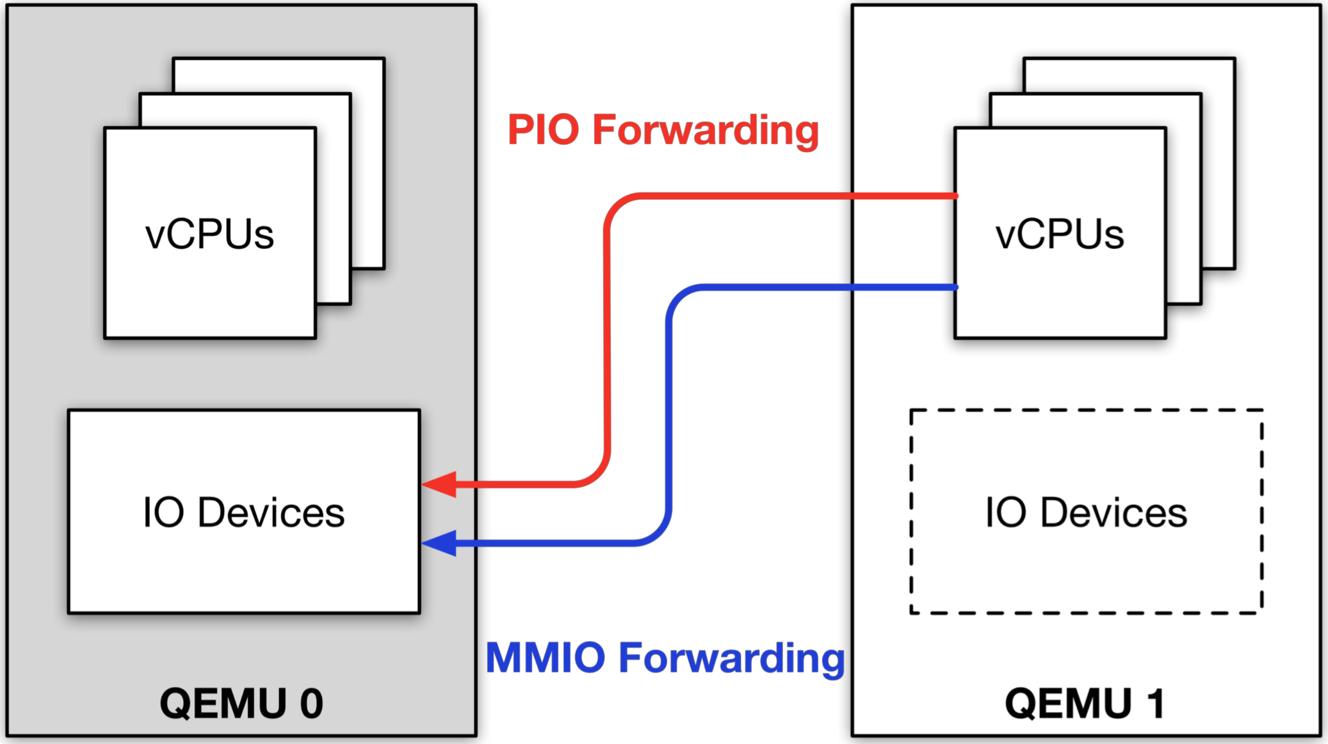
Design: I/O

- I/O Forwarding
  - PIO
  - MMIO



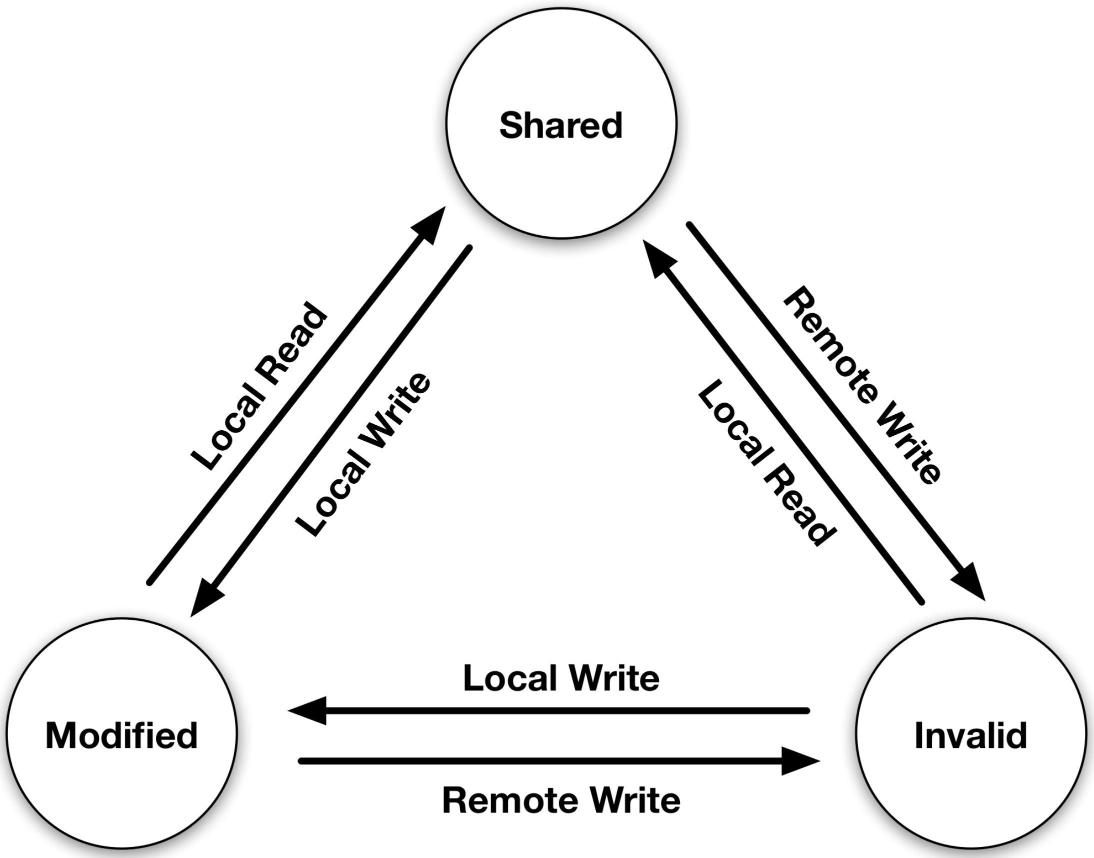
Design: I/O

- I/O Forwarding
  - PCI/PCIe devices
    - Manipulate configuration space by specific PIO
    - MMIO memory region is dynamically configured through configuration space



Design: Memory

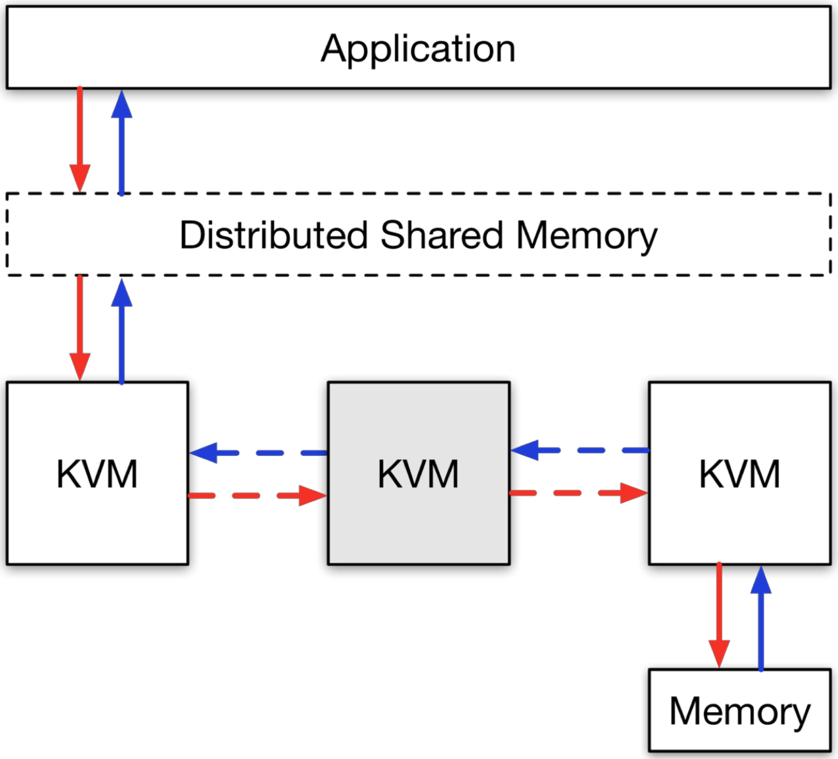
- DSM Protocol
  - Page granularity
  - Modified / Shared / Invalid state, controlled by EPT



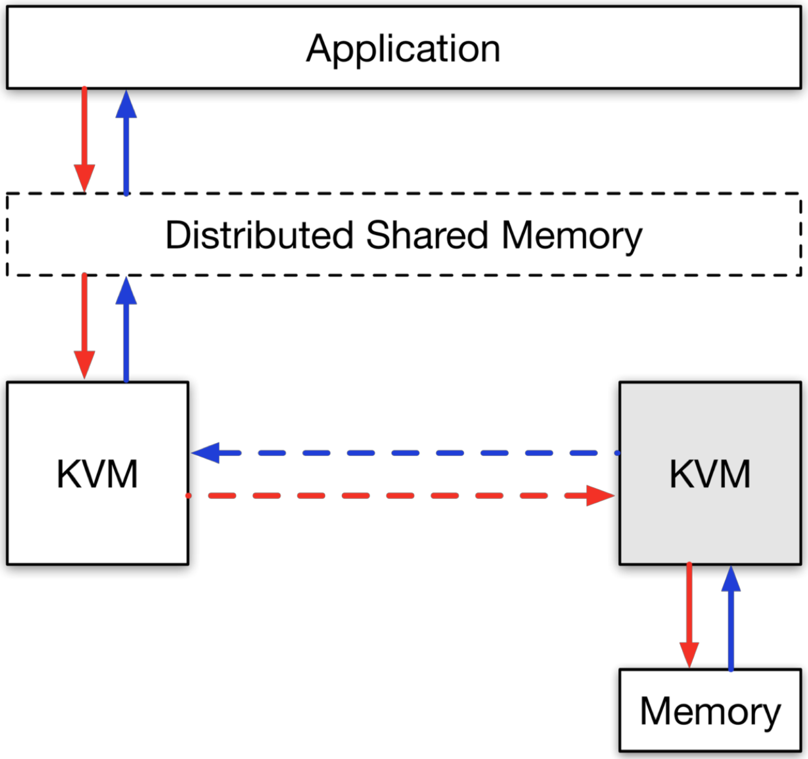
Ref: Kai Li, Paul Hudak. 1989. Memory Coherence in Shared Virtual Memory Systems.

Design: Memory

- DSM Protocol
  - Designate manager to track the owner of pages



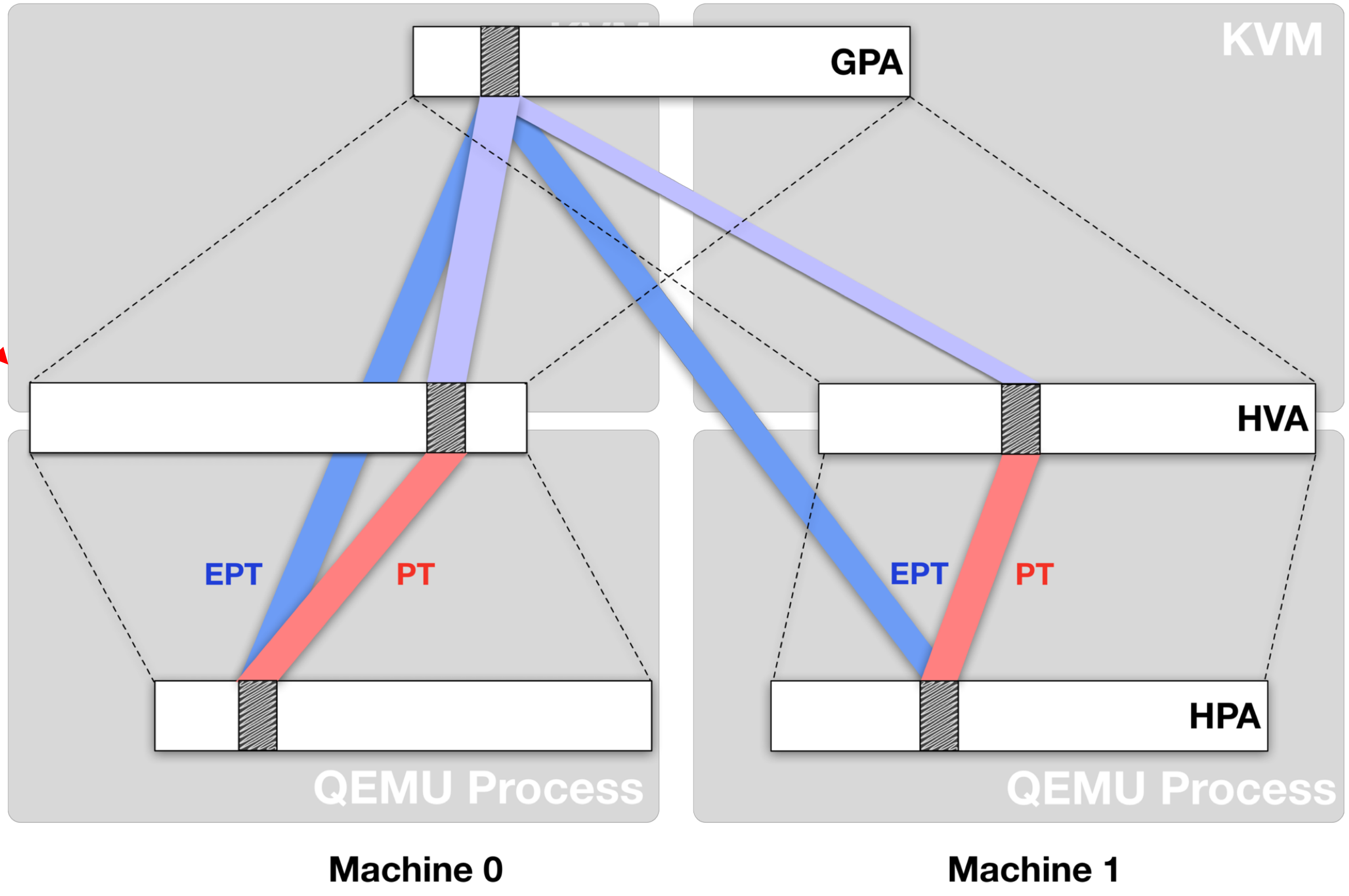
Manager is not owner



Manager is owner

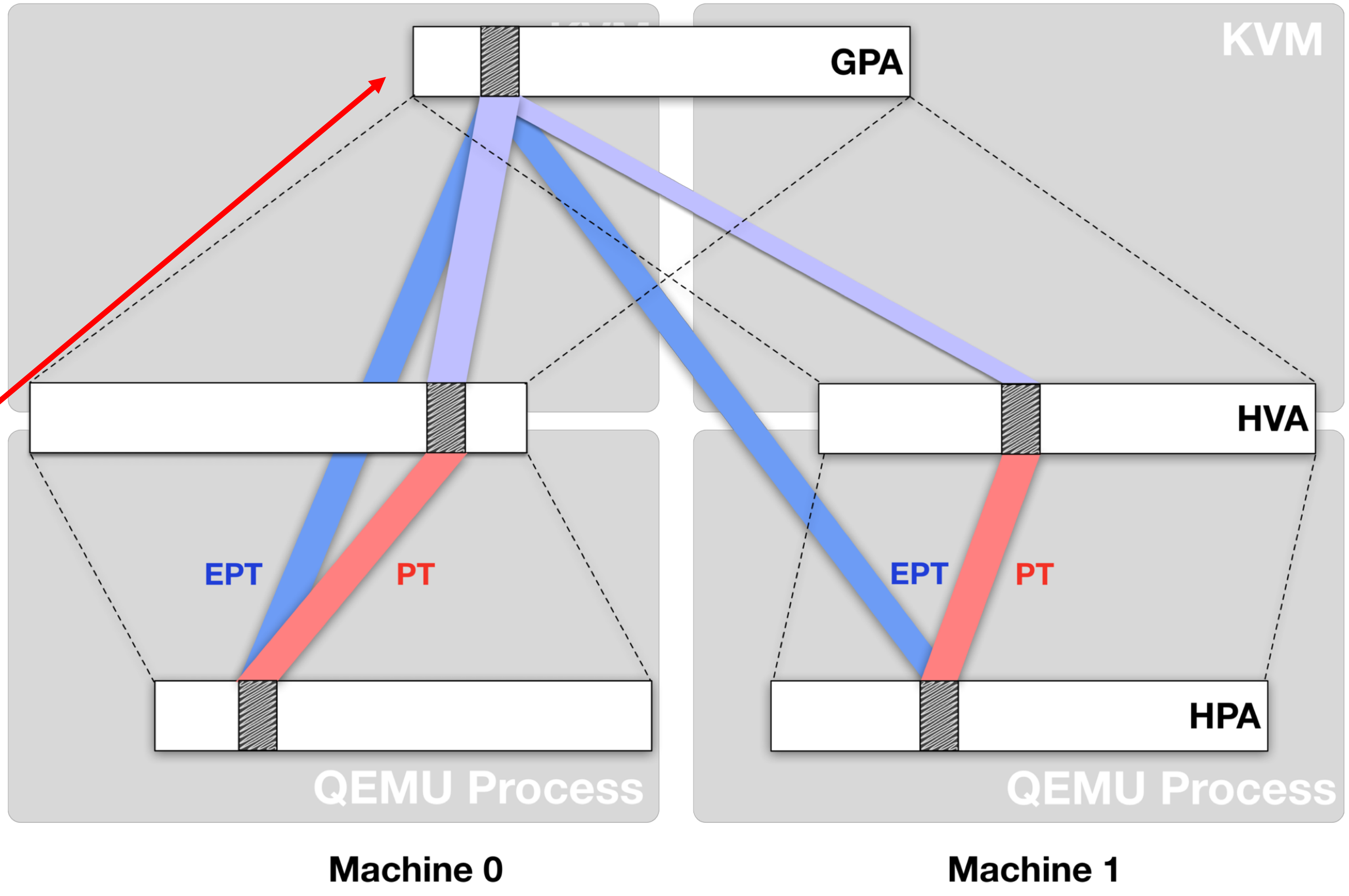
Design: Memory

- Mapping Management
  - Guest memory is allocated in HVA space
  - The only shared memory space between machines is GPA space



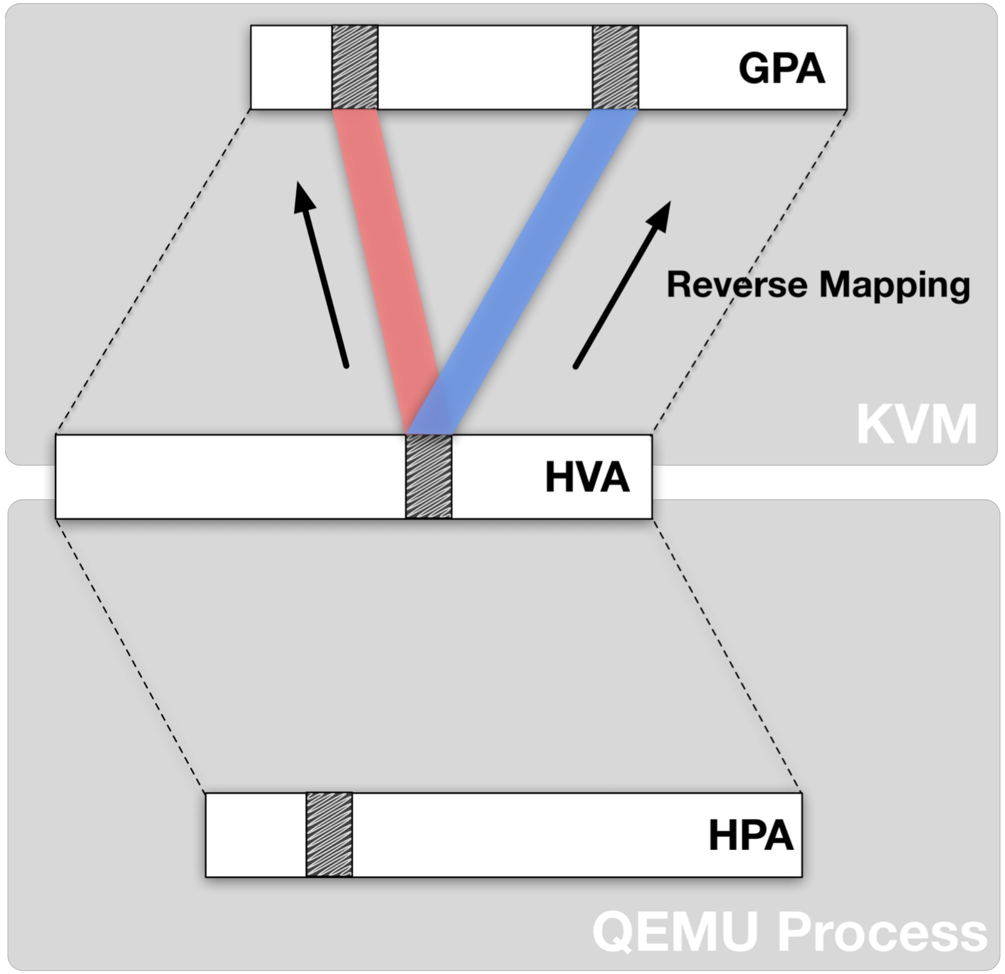
Design: Memory

- Mapping Management
  - Guest memory is allocated in HVA space
  - The only shared memory space between machines is GPA space



Design: Memory

- Mapping Management
  - Maintain state and copyset of pages in HVA space
    - Two or more guest pages mapping to the same host page
  - Maintain reverse mapping from host to guest pages



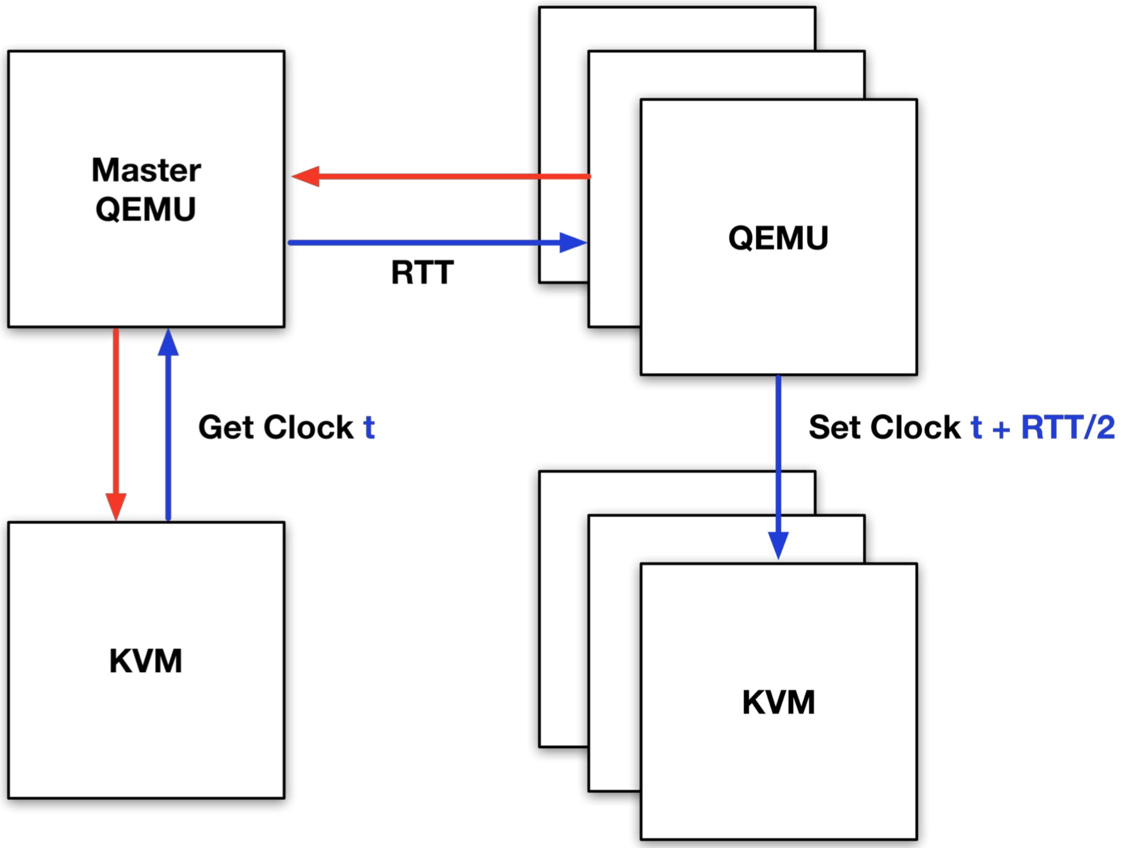
Machine X

Design: Memory

- Memory Accesses Bypassing EPT
    - QEMU and KVM can manipulate guest page directly
    - Without involving the translation provided by EPT
    - Add **pin** and **unpin** operation(DSM_MEMPIN) to DSM
        1. pin
        2. transfer it into the desired state
        3. block all the incoming requests on this page
        4. unpin

Design

- Time Synchronization
  - kvmclock
    - Keep kvmclock value on different vCPUs in sync
    - Other QEMUs query and apply the kvmclock value from master QEMU



Agenda

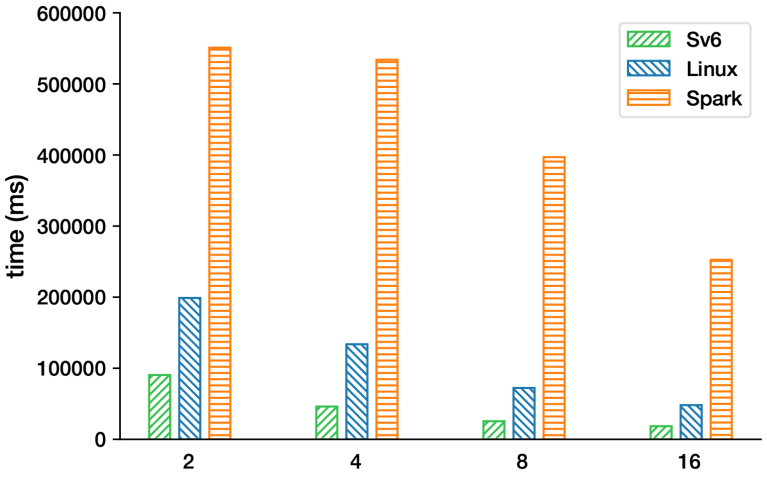
- Demo
- Motivation & Challenge
- Design Detail
  - CPU
  - I/O
  - Memory
- **Performance**

Performance

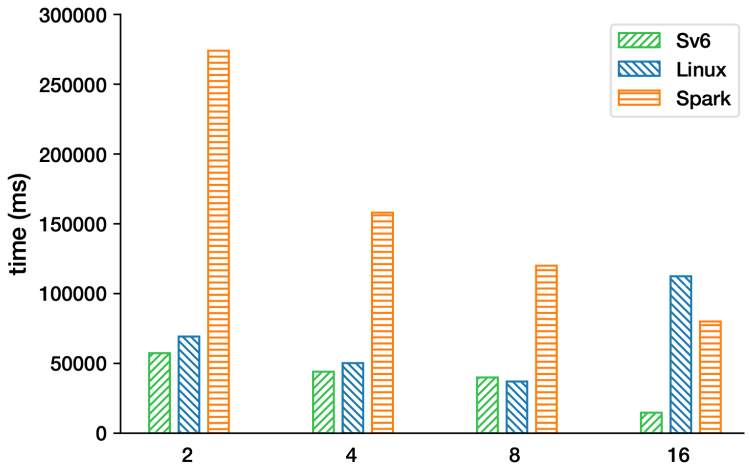
- Experiment Machine Configuration
  - 2 machines
  - 16 core Intel Xeon E5-2620 v4
  - 128GB DRAM
  - Broadcom NetXtreme BCM5720 Gigabit Ethernet (For TCP)
  - ConnectX-3 MCX354A-FCBT 56Gbps InfiniBand (For RDMA)

Performance

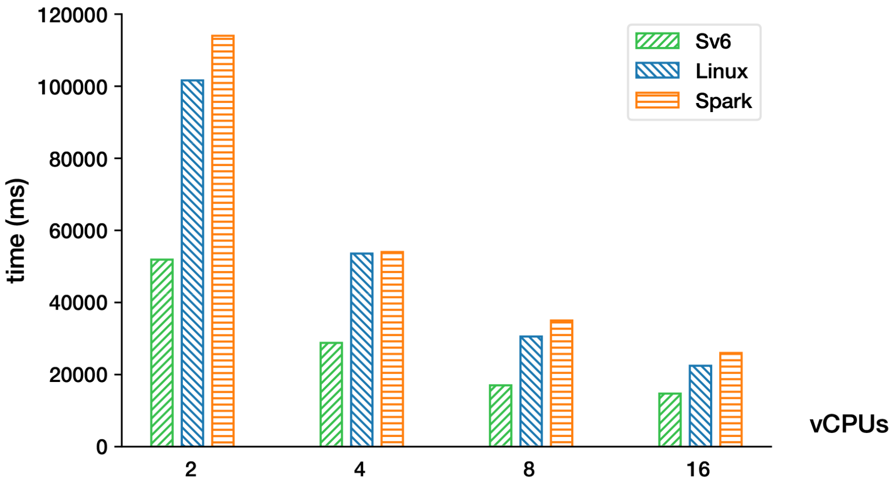
- Measure applications running on sv6 / Linux / equivalent Spark jobs
- Profile the time consumption of each component
    - Bottleneck is **DSM PF**



PI



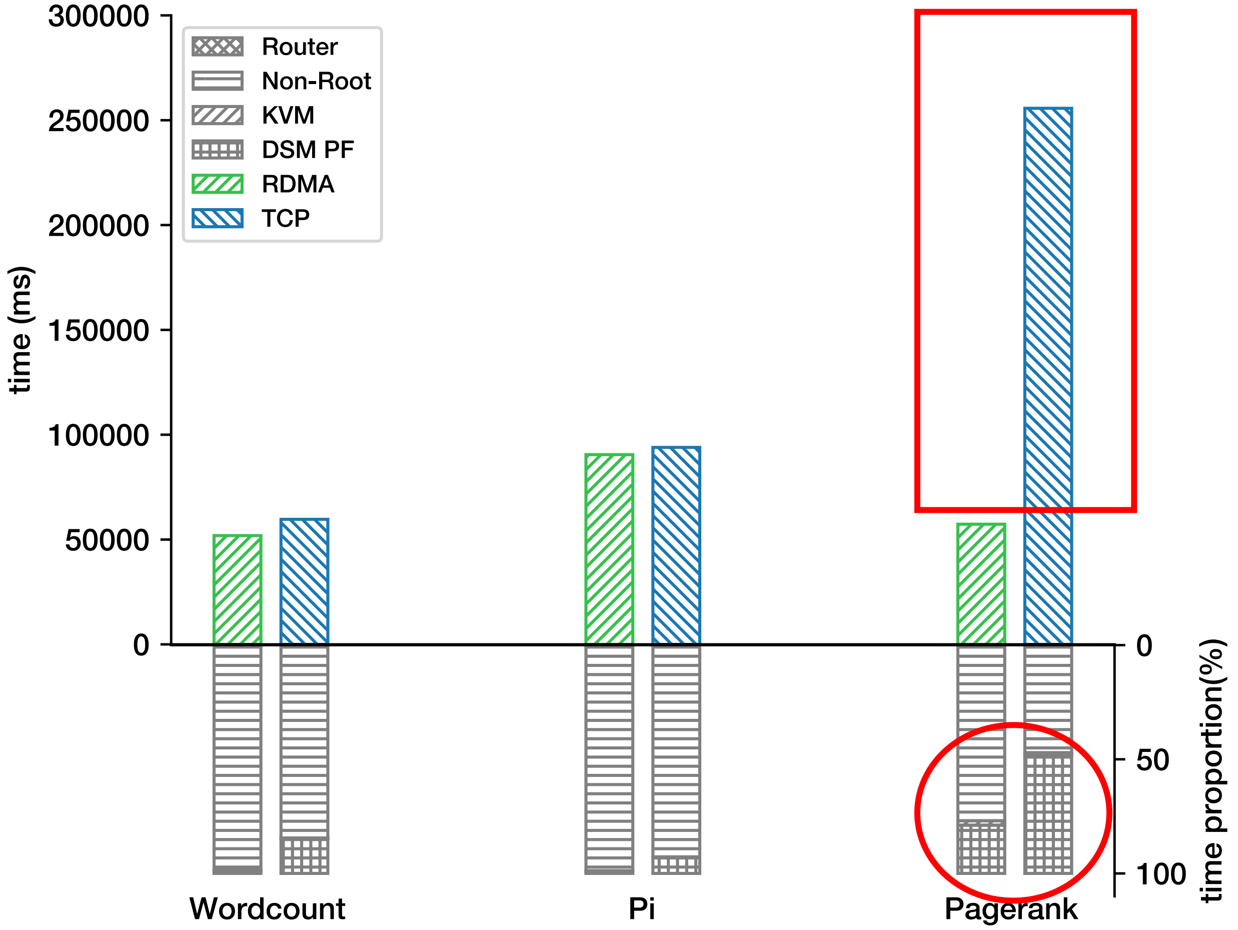
Pagerank



Wordcount

Performance

- Comparison between using RDMA backend and TCP backend
  - RDMA backend reduce the time consumption actually
  - DSM PF benefits from the fast RDMA network



Summary

- We extend QEMU to support running cross machine VM
  - Distributed vCPU
  - Interrupt Forwarding
  - I/O Forwarding
- We implement Distributed Shared Memory for VM by extending KVM
- We evaluate the performance of cross machine VM

Future Work

- Dirty/Access bit support of EPT is disabled
- Support IRQCHIP in KVM
- Distributed I/O devices
- Fault Tolerance

Thanks!

https://github.com/GiantVM/homepage

Q&A