QuartzV: Bringing Quality of Time to Virtual Machines

Sandeep D'souza and Raj Rajkumar
Carnegie Mellon University

Carnegie Mellon

ROSELINE

IEEE RTAS @ CPS Week 2018

A Shared Notion of Time

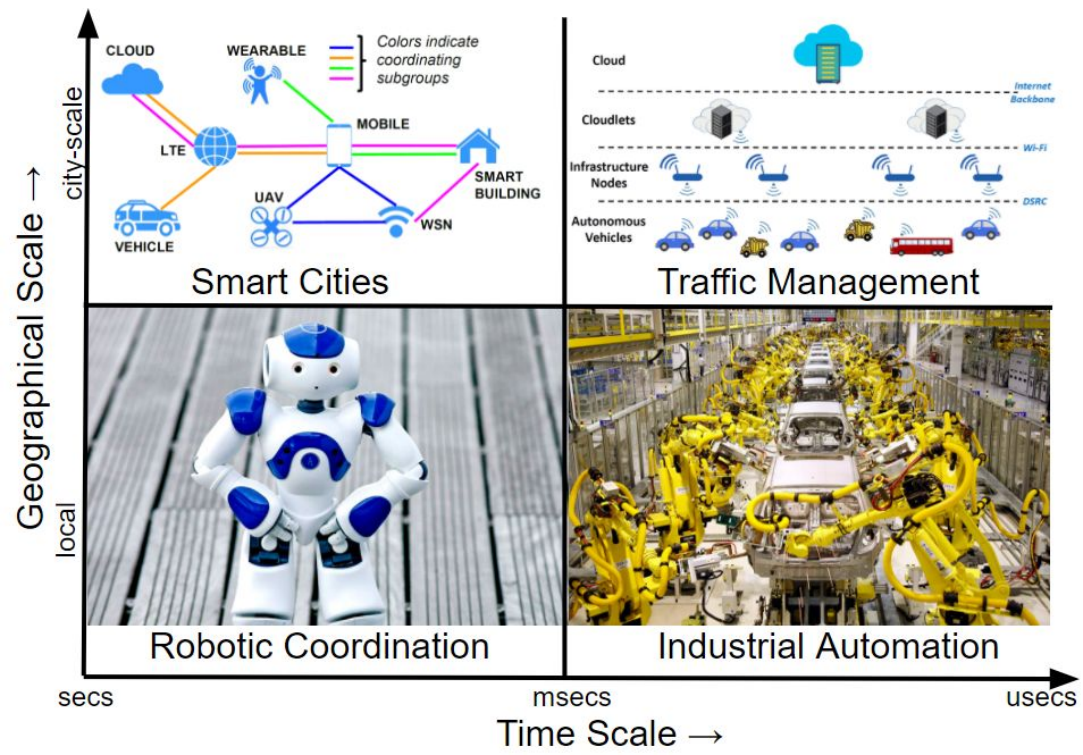
- ***Coordinated*** Actions
- ***Ordering*** of Events

A Shared Notion of Time is *useful*

→ Replace *Communication* with *Local Computation**

*Liskov, *Distributed Computing '93*

Coordination in Space and Time*

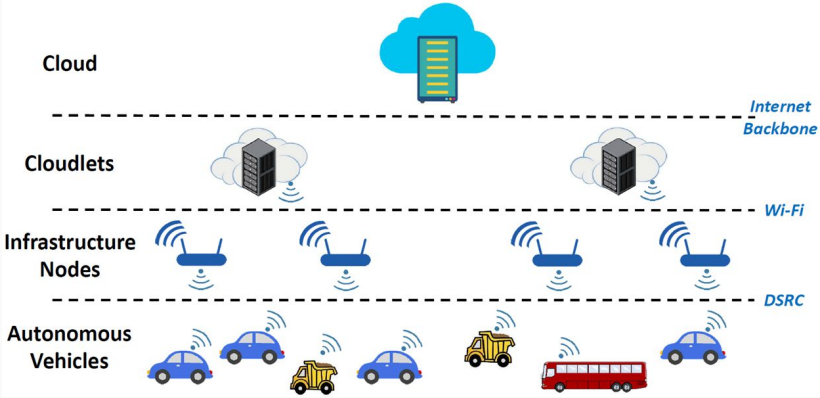


The cloud/edge is *key* to achieve scale

*D'souza et al., *HotCloud '17*

Cyber-Physical Edge-Cloud

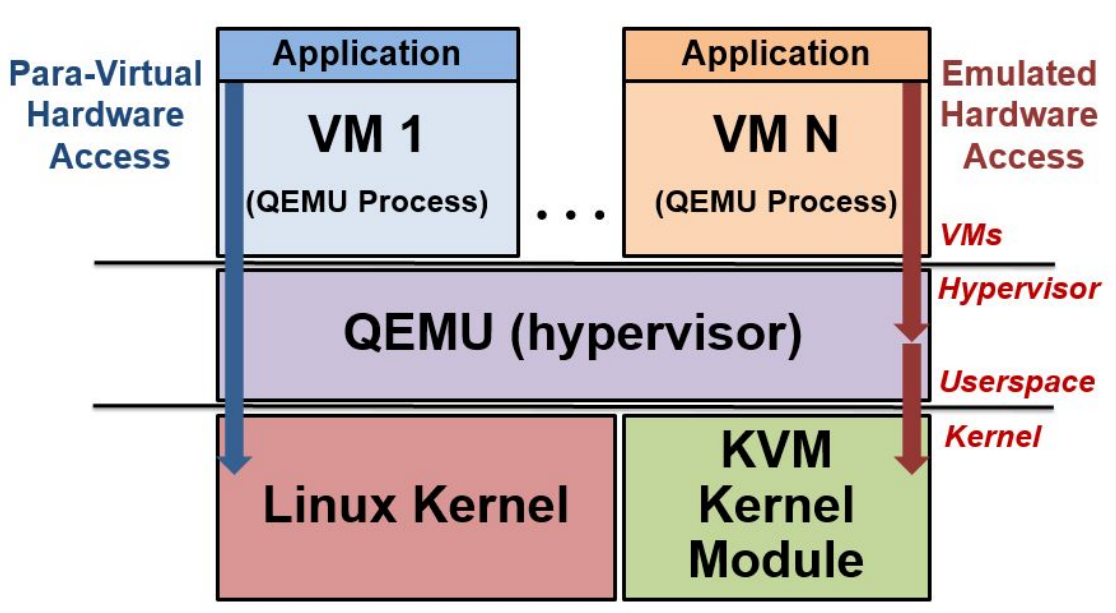
- **The cloud is *key* to enable geographical scaling**
  - data ***storage***
  - host the ***intelligence*** behind CPS
- **Low-latency requirements of CPS**
  - ***Safety-critical*** + ***real-time*** performance
  - A ***hierarchy*** of edge and cloud deployments



Virtualization is *key* for utilizing cloud and edge resources

Virtualization

- OS-level (Containers)
- Hypervisor-based (VMs)
  - *CPU Instructions*
    - Emulation
    - *Hardware-Acceleration*
  - *Peripheral Access*
    - Full Virtualization
    - *Para-Virtualization*



Hardware-Accelerated CPU Instructions + *Para-Virtual* Peripheral Access
→ *near-native performance for VMs*

Outline

- Motivation
- **Background**
  - Quality of Time (QoT)
  - Timelines
  - QoT Stack for Linux
- **QuartzV**
- **Experimental Evaluation**
- **Conclusion**

Quality of Time (QoT)*

- **Quantified**
  - using ***clock parameters***:
    - accuracy, precision, drift….
  - w.r.t a ***reference clock*** (time)
- **Each timestamp has *high-probability* bounds**
  - Timestamp ϵ $\{t-\varepsilon_l, t+\varepsilon_h\}$

The *end-to-end* uncertainty in the notion of time delivered to an application by the system

*Anwar et al., *RTSS '16*

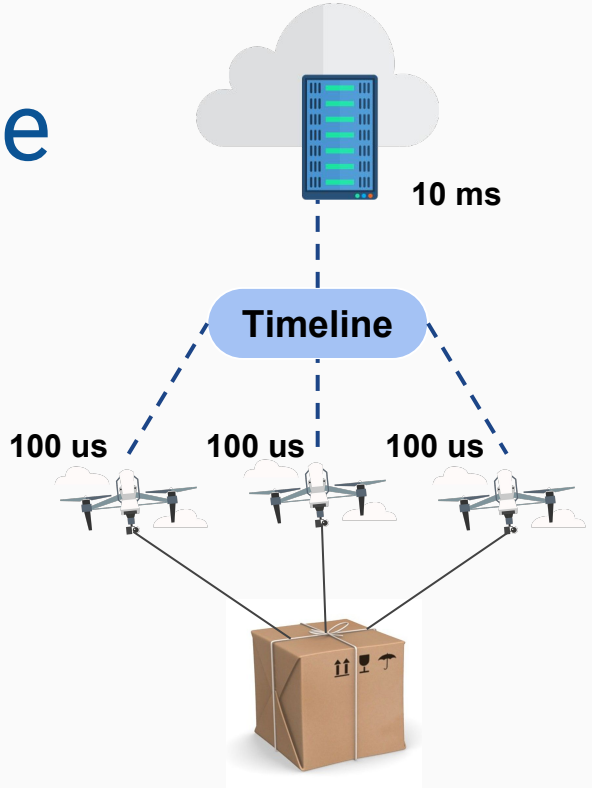
QoT and Fault Detection

- **Failure Scenario:**
  - ***Clock Synchronization*** degrades
  - ***Reported QoT*** must reflect degradation
- **Application-specific *failover* mechanisms**
  - ***Physical*** and ***Analytical*** Redundancy

QoT can enable fault detection
→ *fault-tolerant* coordination in CPS

Enabling Coordination at Scale

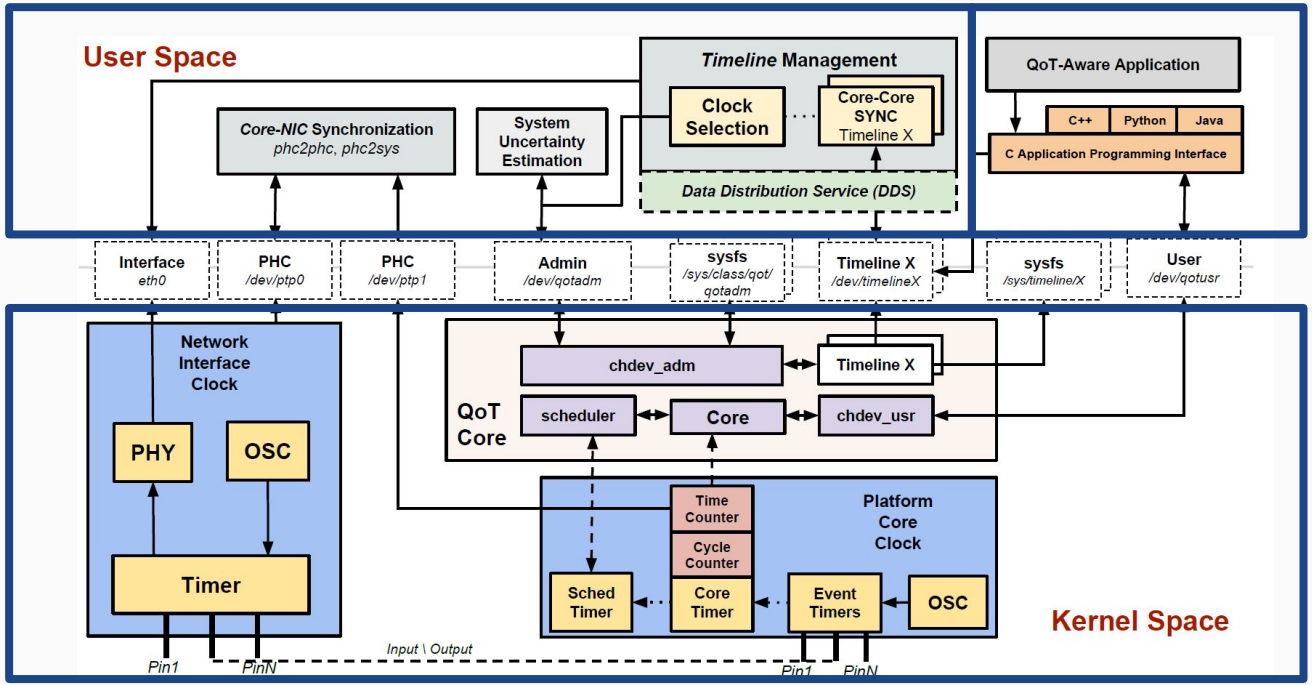
- **Timeline*: *Virtual time reference***
  - ***time-based*** coordination service
  - ***Platform-independent* API**
- ***Coordinated actions*, distributed components**
  - all components ***bind*** to a timeline
  - each ***specifying*** its required QoT



Timelines *abstract* away clock synchronization
→ Applications *specify* QoT requirements, framework *orchestrates* the system

*Anwar et al., *RTSS '16*

QoT Stack for Linux*



Support for ARM and x86 platforms

open source, *modular* implementation, *no change to the Linux kernel*

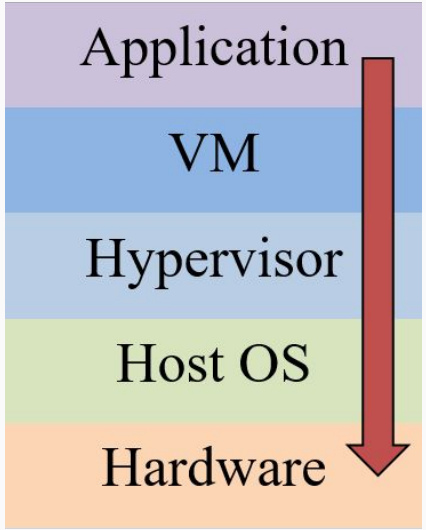
*Anwar et al., *RTSS '16*

Outline

- Motivation
- Background
- QuartzV
  - Challenges
  - Para-Virtual Design & Implementation
  - Full-Virtualization Support
- Experimental Evaluation
- Conclusion

Virtualization and Time

- **Higher clock-read and interrupt latencies***
  - Overhead of additional abstraction layers
- **Delivering application-specific requirements**
  - different ***QoT requirements***
  - different ***time scales***



Providing *near-native* timing performance is a challenge

*Broomhead et al., *OSDI '10*

Quartz-V: Challenges

- **Low-Latency Clock Reads + QoT estimates**
- **Supporting multiple VMs → *multiple timelines***
- **Maintaining VM Isolation**
  - prevent ***malicious VMs*** from affecting correct timing
- **Portability**
  - ***avoiding modifications*** to the kernel, hypervisor & application source

Objective: *near-native* performance while *maintaining* isolation and portability

Quartz-V: Key Ideas

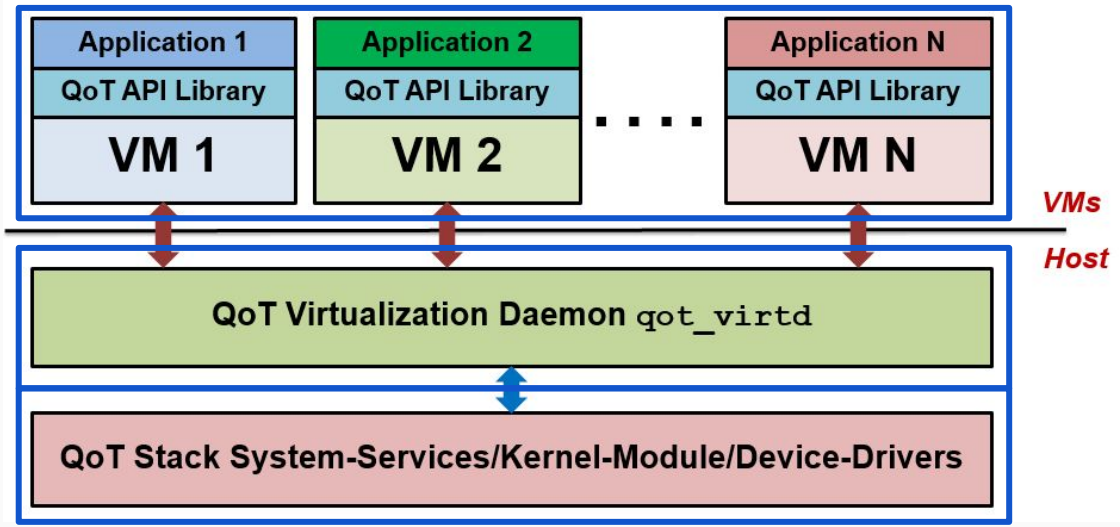
- **Extends the QoT Stack for Linux**
  - QEMU-KVM hypervisor extensions for *Virtual Machines*
- *Para-Virtual* (PV) **approach: "*Time-as-a-service*"**
  - PV guests ***specify*** Timeline and QoT requests
  - Host ***performs*** clock sync + QoT Estimation
    - ***transfers*** the ***sync parameters*** to the guest
  - PV guests ***read*** a **para-virtual** host clock (`KVM-Clock`)
    - ***compute*** the *"timeline"* time using **host-provided** ***parameters***

$$tl_{now} = tl_{last} + tl_{skew} * (core_{now} - core_{last})$$

current time — ***offset*** — ***clock skew*** — ***time passed since the clock was last synchronized***

qot_virtd: QoT Virtualization Daemon

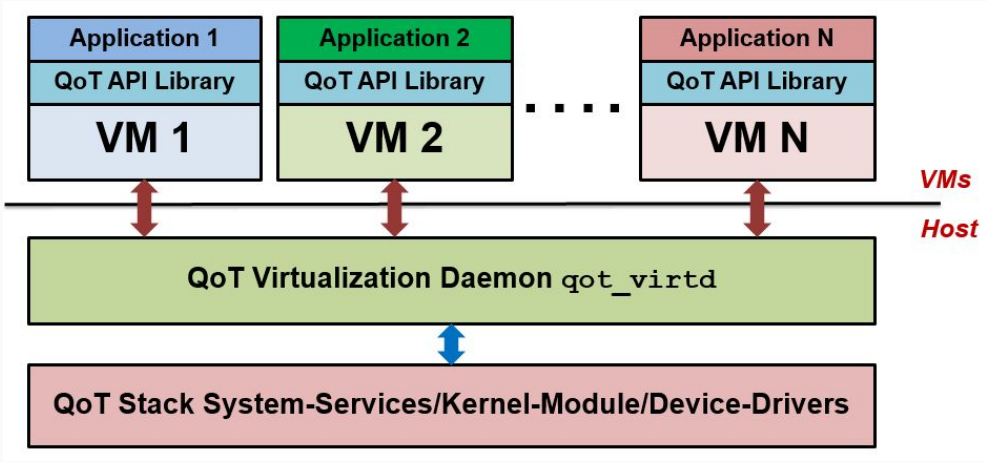
- **`qot_virtd` is the interface between:**
  - QoT-aware ***applications*** in the guest VM, **and**
  - QoT Stack ***system-services/kernel-module*** on the host



`qot_virtd` acts as a *server* for *client* QoT-aware applications in a VM

Specifying Application Requests

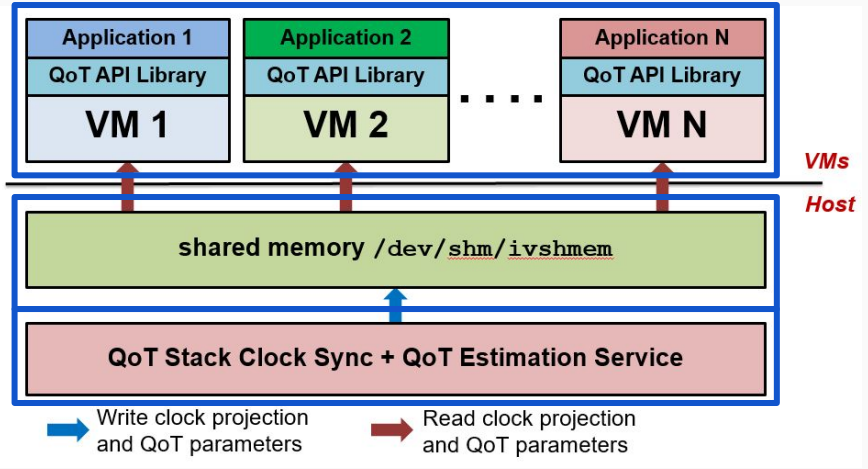
- ***not*** on the application critical path
  - ***reliable,*** need ***not*** be low latency
- `virtio_serial`
  - *bi-directional* serial communication
  - *scalable* across multiple VMs



`virtio_serial` allows application requests to be *specified* to `qot_virtd`
hypervisor mechanism → *no new* changes required

Provide Clock + QoT Parameters

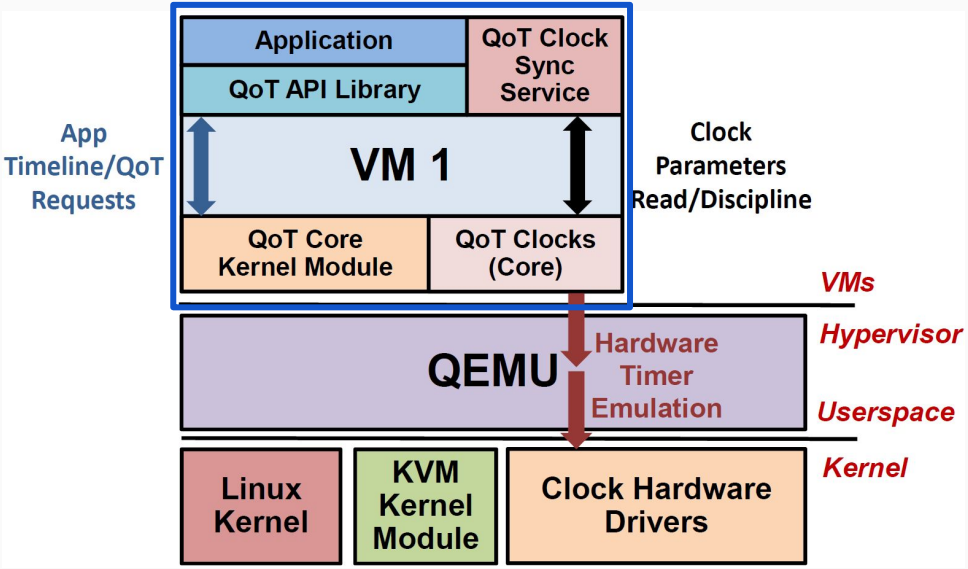
- **Clock parameters *required***
  - to ***compute*** the current time
- **Clock reads** are on the ***critical path***
  - ***reliable*** and ***low latency***
- **Inter-VM *Shared Memory* (`ivshmem`)**
  - *Host clock-sync service* **writes** parameters
  - *Guest Applications* can **read** parameters
    - reading memory is *low latency*
    - VMs have ***read-only*** access, *maintains isolation*



QuartzV provides *near-native* performance while *maintaining* isolation and portability

Fully-Virtual QoT Stack

- **Some VMs/Hypervisors**
  - do not support paravirtualization
- **The *entire* QoT Stack inside a VM**
  - ***Fully-Virtual*** deployment
  - clock sync achieves ***lower accuracy***
  - clock reads have ***higher latency***



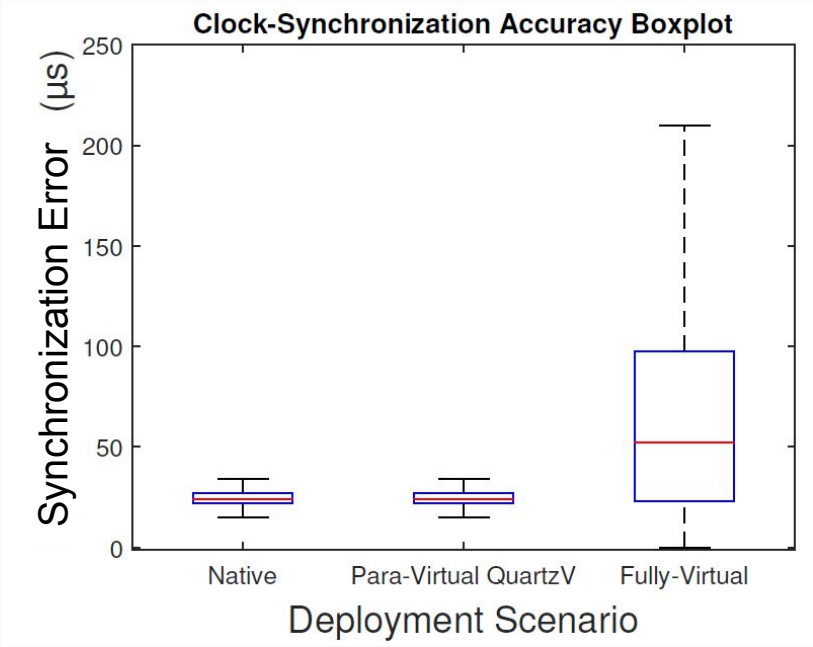
The *Fully-Virtual* QoT Stack can provide *full functionality*, but *lower performance* w.r.t. QuartzV

Outline

- Motivation
- Background
- QuartzV
- Experimental Evaluation
  - Performance Evaluation
  - Scalability Evaluation
  - Prototype Industrial Robotics Application
- Conclusion

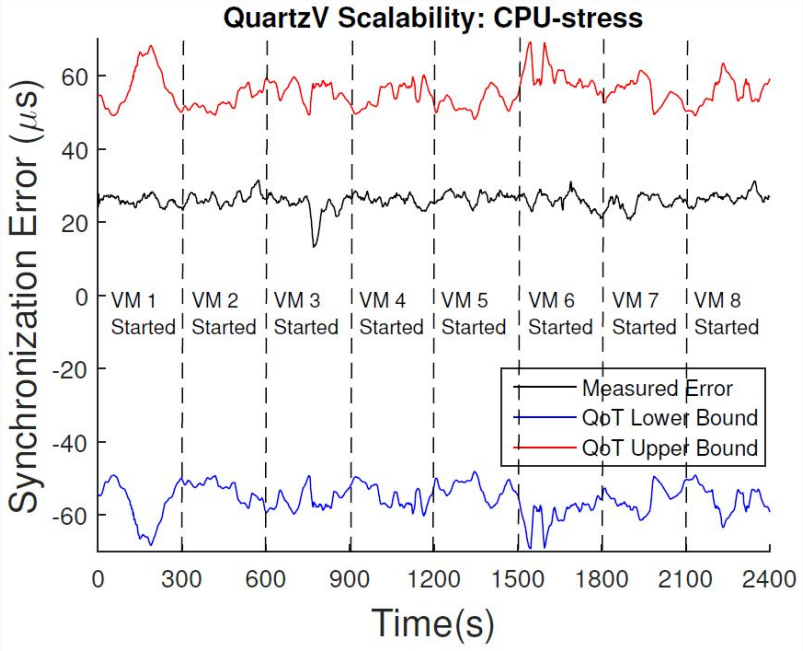
Clock-Synchronization Accuracy

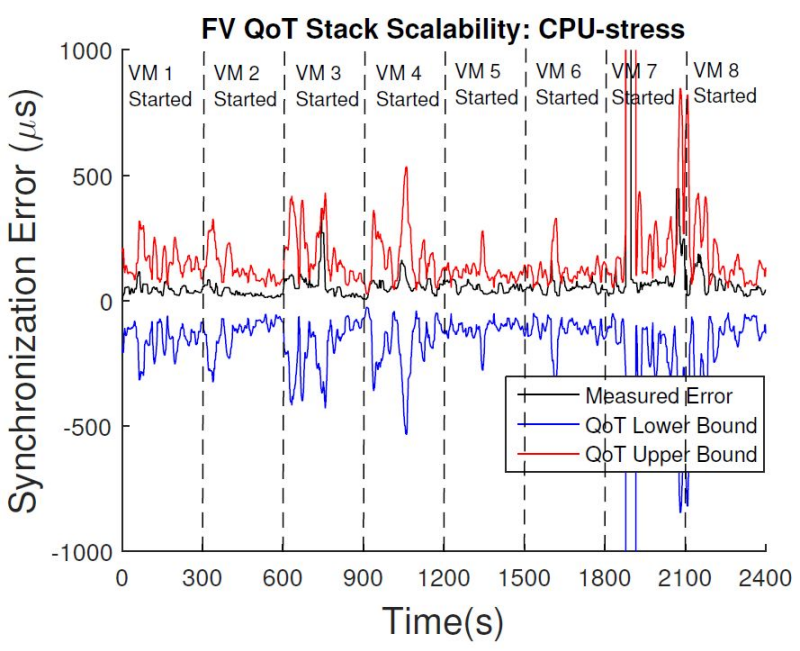
- **Comparison using PTP (`qot_ptp`)**
  - 8-core i7 CPU, with 16GB RAM
  - Clock Evaluation Test-Bed
- ***QuartzV***
  - yields ***near-native*** accuracy
  - clock sync on the host
- ***Fully-Virtual QoT Stack***
  - yields ***lower*** accuracy
  - clock sync in the VM
  - ***overhead*** of virtualized network stack



QuartzV can provide *near-native* clock-sync accuracy

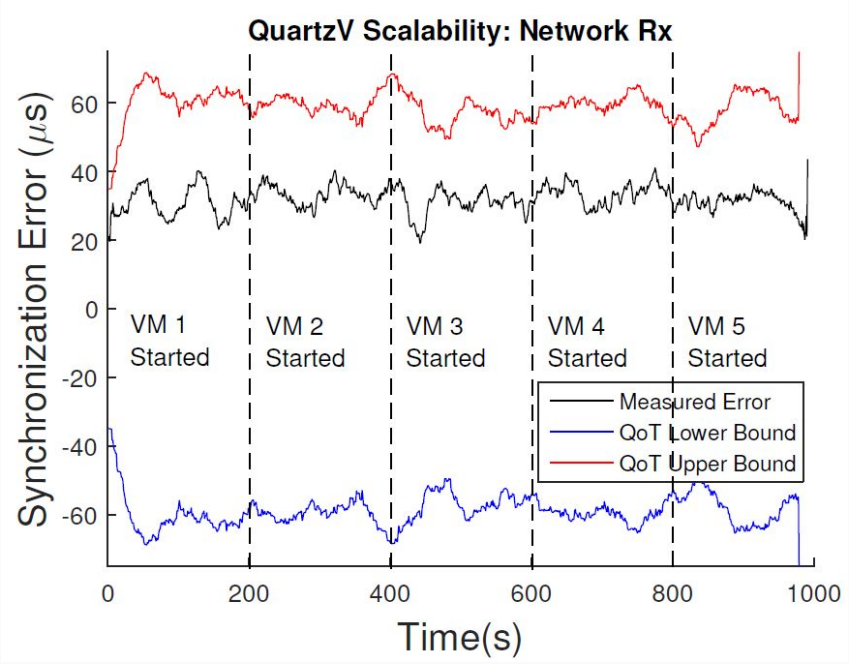
CPU-stress Scalability Results

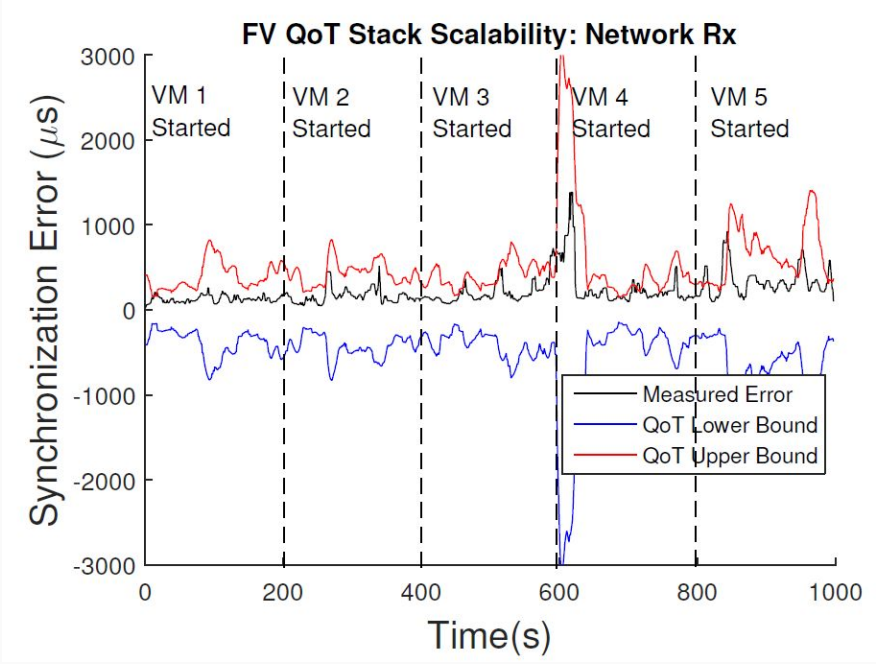




QuartzV is more *resilient* to CPU-intensive applications

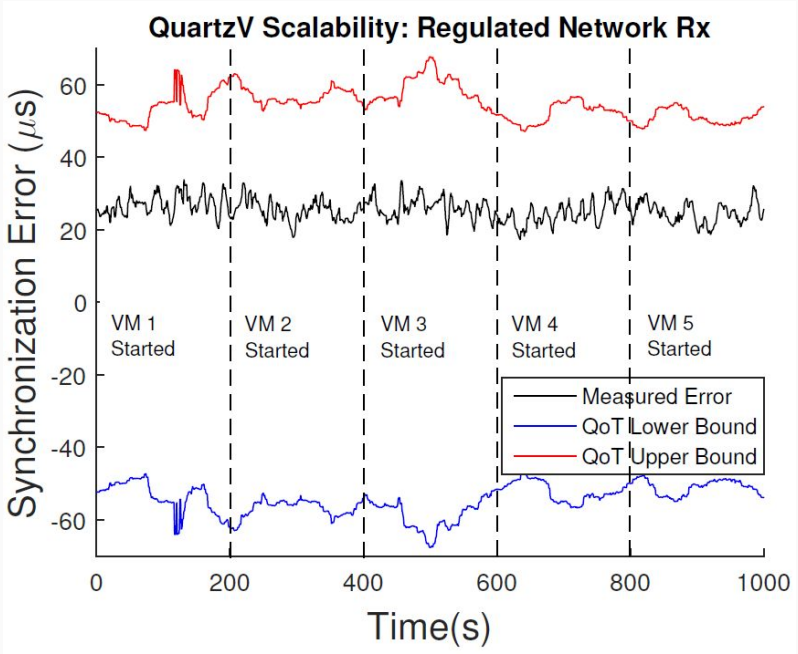
Network-stress Scalability Results

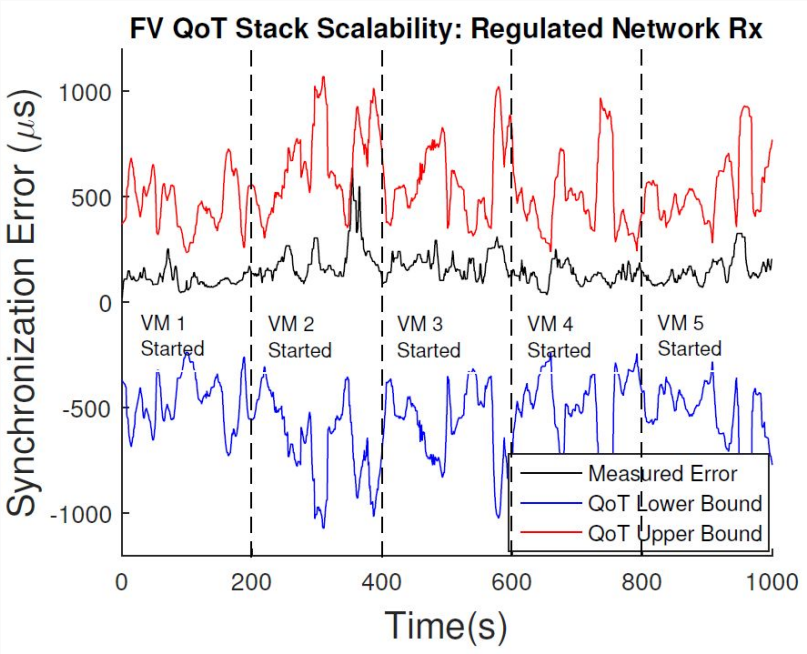




QuartzV is more *resilient* to *malicious* network-intensive VMs

Network-Regulation Scalability Results

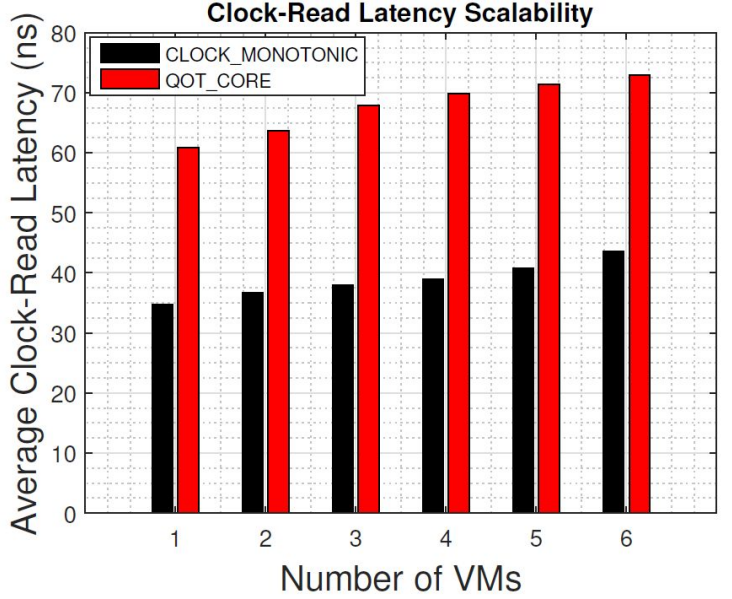




Bandwidth Regulation can *prevent* malicious network-intensive VMs from degrading clock-synchronization accuracy

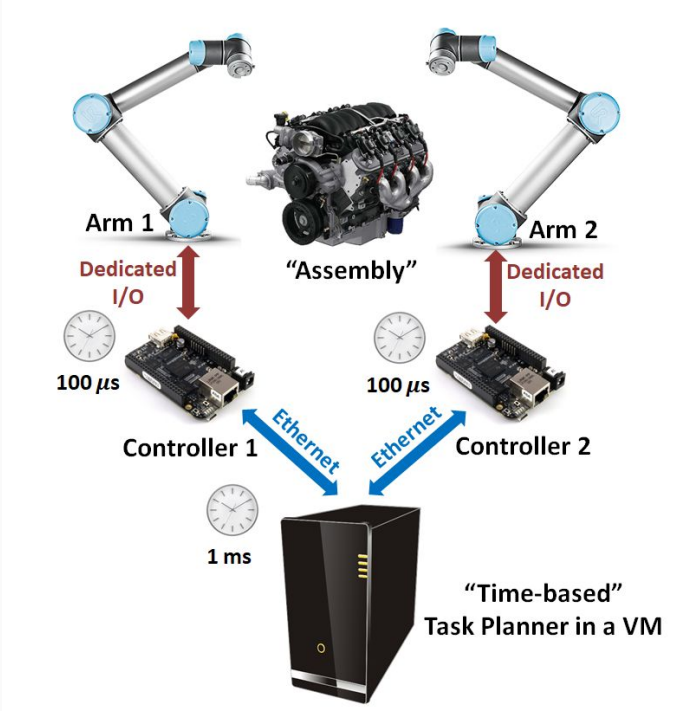
QuartzV: Clock-Read Latency Scalability

- **Concurrent VMs performing clock reads**
- **QoT Clock v/s CLOCK_MONOTONIC**
  - ***higher*** latency,
  - due to ***applying*** projection parameters,
  - and QoT calculations
- ***Slight*** **increase in latency with new VMs**
  - contention in ***reading*** the hardware timer
  - same trend for both clocks



Shared Memory supports *simultaneous* read-only access
→ *Bottleneck-free* clock-read implementation

Coordinated Industrial Robotics



ROS-Gazebo Robot Physics Simulation

Industrial Edge Controllers

QuartzV-enabled Industrial Edge Cloudlet

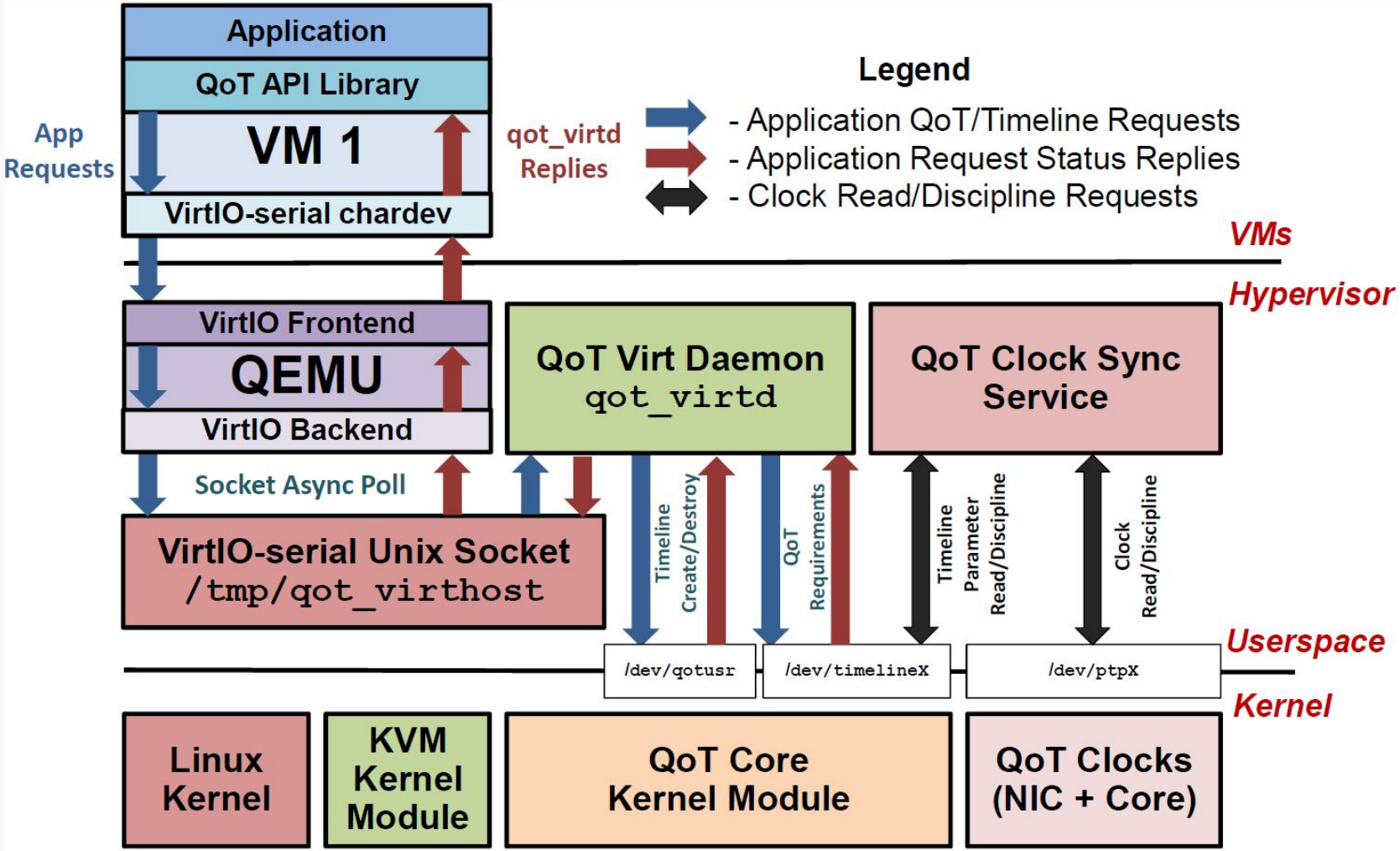
Conclusion and Future Work

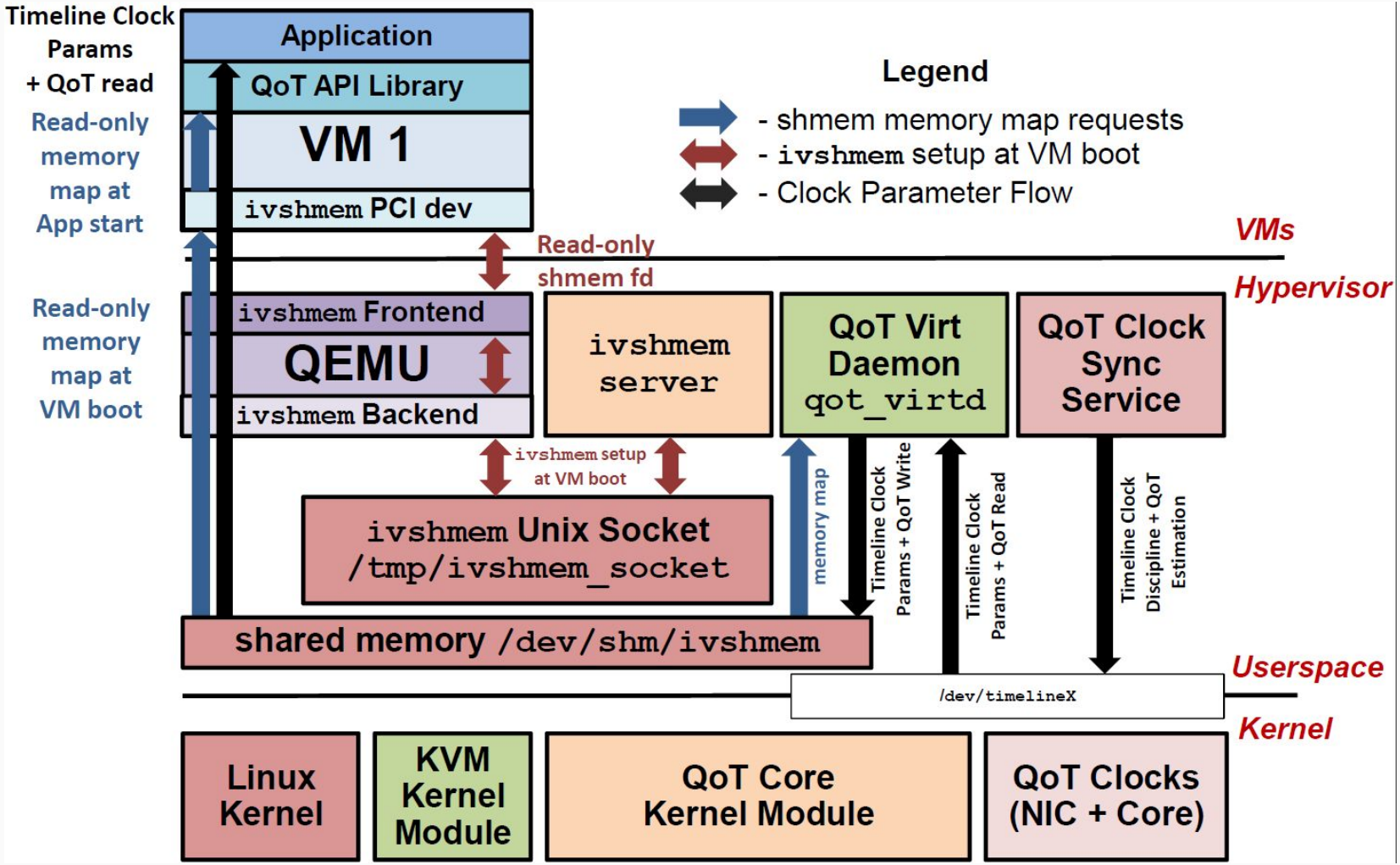
- **Geo-Distributed CPS: "*Coordination at scale using time*"**
- **QoT-awareness *enables* Fault Detection**
- ***Quartz-V:* Adds QoT-awareness to VMs**
  - uses the ***para-virtual*** approach
    - ***delivers near-native*** timing performance
    - ***maintains isolation*** and ***scales*** to multiple VMs
    - ***resilient*** to malicious resource-intensive VMs
- ***Fully-Virtual QoT Stack*** for hypervisors/VMs with **no** para-virtual support
- **Future Work**
  - ***Building a Geo-Scale QoT-based CPS Coordination Framework***
- **Open source:** *https://bitbucket.org/rose-line/qot-stack/src*

QuartzV can *enable* hosts of new geo-distributed coordinated Cyber-Physical Systems

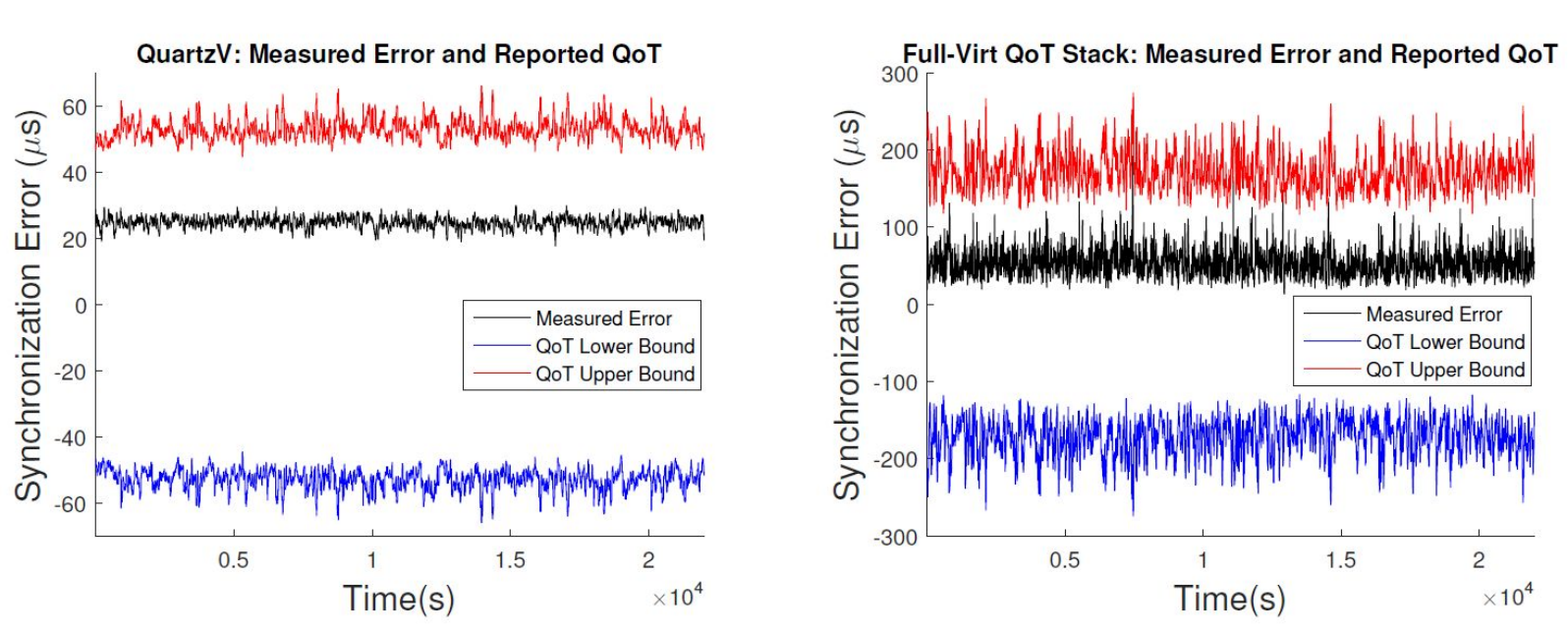
Thank You ! Questions ?

Sandeep D’souza
sandeepd@andrew.cmu.edu


Application
QoT API Library
VM 1
VirtIO-serial chardev
App Requests
qot_virtd Replies
Legend
- Application QoT/Timeline Requests
- Application Request Status Replies
- Clock Read/Discipline Requests
VMs
Hypervisor
VirtIO Frontend
QEMU
VirtIO Backend
QoT Virt Daemon
qot_virtd
QoT Clock Sync Service
Socket Async Poll
VirtIO-serial Unix Socket
/tmp/qot_virthost
Timeline Create/Destroy
QoT Requirements
Timeline Parameter Read/Discipline
Clock Read/Discipline
Userspace
/dev/qotusr
/dev/timelineX
/dev/ptpX
Kernel
Linux Kernel
KVM Kernel Module
QoT Core Kernel Module
QoT Clocks (NIC + Core)


Timeline Clock Params + QoT read
Read-only memory map at App start
Read-only memory map at VM boot
Application
QoT API Library
VM 1
ivshmem PCI dev
ivshmem Frontend
QEMU
ivshmem Backend
Legend
- shmem memory map requests
- ivshmem setup at VM boot
- Clock Parameter Flow
Read-only shmem fd
VMs
Hypervisor
ivshmem server
QoT Virt Daemon qot_virtd
QoT Clock Sync Service
ivshmem setup at VM boot
ivshmem Unix Socket /tmp/ivshmem_socket
memory map
Timeline Clock Params + QoT Write
Timeline Clock Params + QoT Read
Timeline Clock Discipline + QoT Estimation
shared memory /dev/shm/ivshmem
Userspace
/dev/timelineX
Kernel
Linux Kernel
KVM Kernel Module
QoT Core Kernel Module
QoT Clocks (NIC + Core)

QoT Estimation Results



Estimated QoT (99.999% confidence)
bounds **the measured clock-synchronization error**