Real-Time KVM for the Masses
Real-Time KVM for the Masses

Agenda

Motivation & requirements

- Reference architecture
- Compute node setup
- Open Stack adaptations
- Summary & outlook
Real-Time Virtualization Drivers

• Communication systems
  (media streaming & switching, etc.)
• Trading systems
  (stocks, goods, etc.)
• Control systems
  (industry, healthcare, transportation, etc.)

=> Consolidation
=> Hardware standardization
=> Simpler maintenance
=> Fast fail-over

Images: Ethernet switch by Ben Stanfield, licensed under CC BY-SA 2.0, stock market by Katrina.Tuliao, licensed under CC BY 2.0
Real-Time KVM is working!

Can I have it in the cloud?
Real-Time Clouds?  
No Problem!  

Oh, you wanna do I/O as well...
Real-Time Connectivity Required
Realistic Deployments

• **Requirement:** Fast enough links to close loops in time
  • Data acquisition (physical world input)
  • Transfer to VM
  • Data processing (← in VM on RT-KVM)
  • Transfer back
  • Data application (physical world output)

• **That means**
  • Private cloud / data center / server cluster close to physical process
  • RT VMs will require access to special networks
    • Isolated standard networks
    • Real-time Ethernets
    • Field buses
Confining the Real-Time Scope

- No QEMU in the loop (feasible but much harder)
- No RT disks (no use case yet, non-deterministic backends)
- I/O via Ethernet (common denominator)
- No device pass-through (feasible but complex)
- No live migration while RT-operational (out of reach so far)

- The reduced RT bill of material
  - RT CPUs
  - RT network
Management Layers

• **Moving from the lab...**
  • Hand-crafted deployments & starter scripts
  • Individual hosts
  • Some dozen VMs per host

• **...into the data center**
  • Hundreds of VMs, both RT and non-RT
  • Many networks, also both RT and non-RT
  • Flexible management and accounting models

• **Cloud-grade, RT-capable managements stack required**
  => OpenStack
  • Broadly used for private clouds
  • Good integration with KVM
Real-Time Network Access

• Options
  • Emulation
  • Pass-through
  • Para-virtual devices => virtio

• Need for RT data plane
  • vhost-net: in host kernel
  • vhost-user: in separate userspace process

• vhost-user enables more RT tuning
  • DPDK-based switch/router
  • Aggressive polling on interfaces, less event signaling
  • Only irqfd (eventfd) from vhost process to vCPU thread
Reference Architecture (with Networking)

RT Switch/Router

RT Guest

vNIC

vhost-user

EMU

openstack

Nova

Controller Services

PREEMPT-RT with KVM

pNIC

Real-time CPUs (isolated)

Best-effort CPUs

Compute Node Hardware

Separate HW or in VM

DPDK

Real-time CPUs (isolated)

Best-effort CPUs

Compute Node Hardware

Separate HW or in VM
Compute Node Setup

- PREEMPT-RT as host kernel
  - Configuration and tuning according to https://rt.wiki.kernel.org
  - Tune power management at kernel and also BIOS-level
  - See also Rik van Riel's slides (KVM Forum 2015)
- Set up isolcpus for 2 sets
  - vCPU threads
  - RT switch data plane threads
- Sufficient non-isolated CPUs required
  - Management processes & threads
  - QEMU event threads
- We use rcu_nocbs == isolcpus so far
  (but not nohz_full – found no relevant impact on worst-case latency)
Compute Node Setup (2)

• Think about RT thread throttling
  /proc/sys/kernel/sched_rt_period_us
  /proc/sys/kernel/sched_rt_runtime_us
  • May suspend busy RT guests
  • But infinitely looping RT guests can starve the host!

• isolcpus does not affect IRQ affinities
  • Needs fine tuning via script and/or irqbalanced

• Even more tuning feasible...
  • But... do your guests need really this?
Simplifying the Setup

- **Bad news:** Still lots of tuning...
- **Good news:** Can be replicated to similar hosts
- **Better news:** There is a tooling framework!
  - [https://github.com/OpenEneaLinux/rt-tools.git](https://github.com/OpenEneaLinux/rt-tools.git)

  `partrt` - Create real time CPU partitions on SMP Linux

  **Usage:**
  
  `partrt [options] <cmd>`
  `partrt [options] create [cmd-options] [cpumask]`
  `partrt [options] undo [cmd-options]`
  `partrt [options] run [cmd-options] <partition> <command>`
  `partrt [options] move [cmd-options] <pid> <partition>`
  `partrt [options] list [cmd-options]`

  - Uses cgroups + various tricks, avoids isolcpus (=> no reboot)
  - Pending full evaluation, seems reusable so far
RT-KVM Control via libvirt

- libvirt only executing higher layers' commands, no own policies
- All required controls upstream since 1.2.13
- For RT-vCPUs
  - Pinning
  - Scheduling parameters setting (policy, priority)
  - Memory locking
- For RT networks
  - QEMU settings to allow sharing of guest RAM with vhost-user process
  - Connecting VM NICs to specific vhost-user ports (identification via socket path)
OpenStack Support for Real-Time – Nova Compute

• Several pieces already available
  • vCPU pinning
  • pCPU dedication
• RT Blueprint under discussion (https://review.openstack.org/#/c/139688)
  • Introduces flavor property `hw:cpu_realtime`
  • Allows tagging of instances and images
  • Requires `hw:cpu_policy = dedicated`
  • Selects
    • QEMU memory locking
    • vCPU thread policy & priority tuning
• Deficits
  • Hard-coded and inappropriate policy/priority (RR, prio 1)
  • 2nd CPU mask required to differentiate between RT and non-RT pCPUs
Real-Time Nova Compute Status

- **Patches by Sahid Ferdjaouï, Red Hat**
  - Implements current blueprint over git master

- **Not accepted for Liberty**
  - Blueprint needs to be merged first but window already closed
  - New target: Mitaka

- **Currently integrating Sahid's patches into our deployment**
  - Plan to come up with extensions to blueprint and code
OpenStack Support for Real-Time – Neutron Networking

- If Neutron shall manage IP assignment for RT networks – all done
- But RT networks tend to be special
  - Network addresses managed by guests or externally
  - Possibly no TCP/IP at all
=> new network type required
- Neutron patches work in progress @Siemens
  - Introduce “unmanaged” networks (IP-free, no DHCP, ... )
  - Agents on compute nodes will report connectivity
    (availability of specific physical networks)
Results?

```c
void get_measurements(void);
```
Summary & Outlook

• Simplify real-time for data centers & similar setups
  • Standardize setup of basic RT scenarios
  • Make RT VMs manageable and accountable

• Full RT stack of KVM & OpenStack feasible
  • Baseline: PREEMPT-RT
  • Standard QEMU & libvirt
  • Patches for Nova and Neutron required
  • Compute node tuning remains improvable

• Future work
  • RT PCI device assignment (challenge: IRQ management)
  • Compute node setup using rt-tools/partrt
  • RT device emulation (requires reworked QEMU patches)
Thank you!

Jan Kiszka <jan.kiszka@siemens.com>