Real–Time Cloud

FOSDEM 2016
Brussels / 30 & 31 January 2016

Henning Schild

Siemens
henning.schild@siemens.com

Motivation

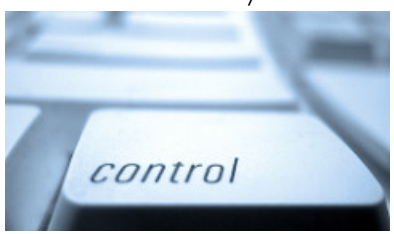
- Communication systems
  - media streaming, switching
- Trading systems
  - stocks, goods
- Control systems
  - industry, healthcare, transportation

All in the Cloud

- Consolidation
- Hardware standardization
- Simpler maintenance
- Fast fail–over

wiertz / CC BY

125992663@N02 / CC BY



fhashemi / CC BY

134465805@N06 / CC BY

Starting point

- we know RT–VMs with KVM are possible
- Preempt–RT host
- start them with hand–crafted scripts
- I/O (NICs) through device assignment
- a few hosts with a couple of VMs each

Does not scale!

- scripts are hard to maintain
  - manual resource assignment
  - error prone
- #NICs limits #VMs
- ...

→ **We want that in the Cloud!**

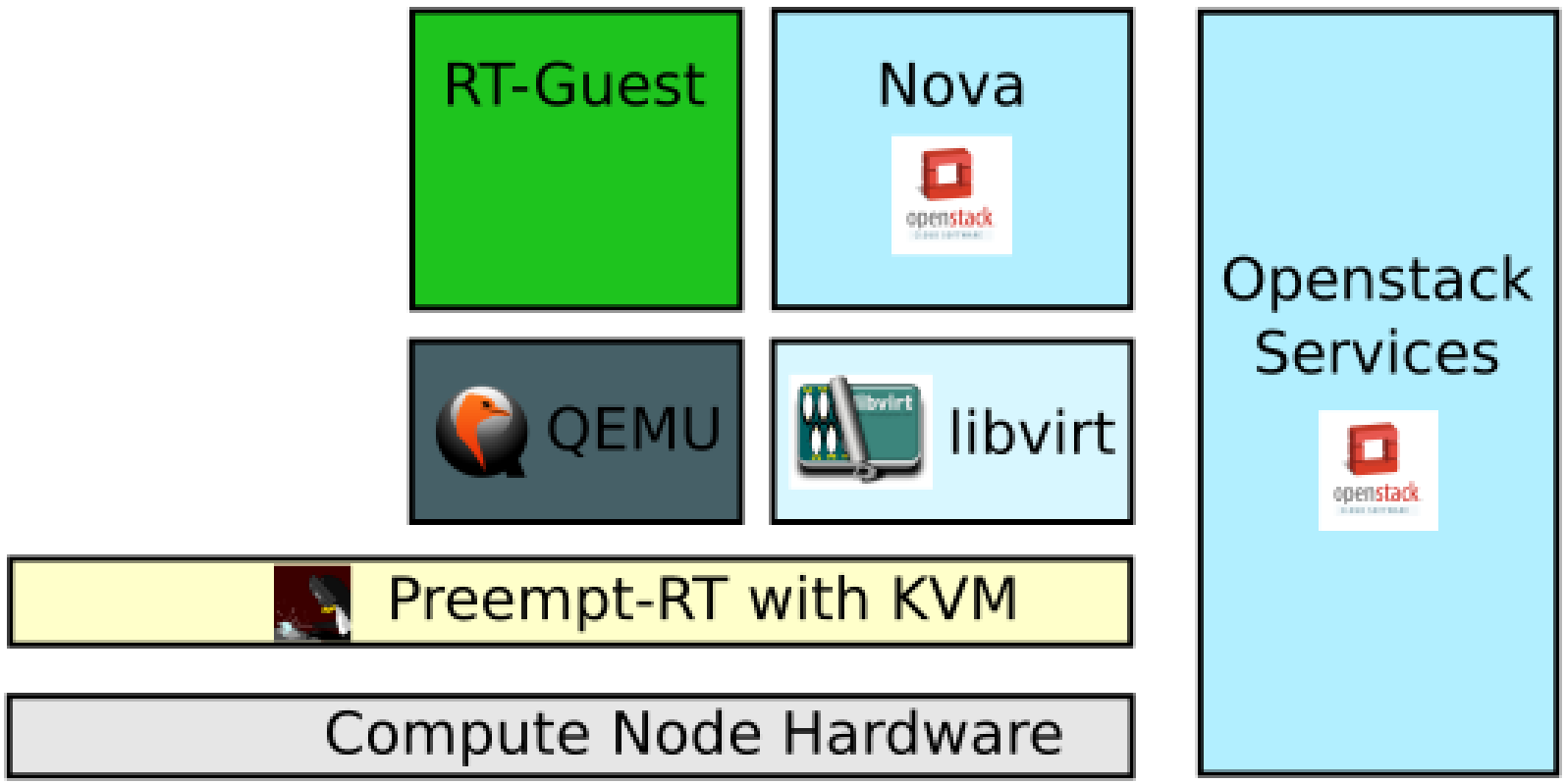
Target

- hundreds of VMs, both RT and best–effort
- many networks, again RT and non–RT
- local deployment needs to be possible
  - not somewhere far away
  - close to the process that should be controlled
  - it is all about CPU + networking latency
- flexible management, multiple users, accounting etc.

Cloud–grade, RT–capable VM management stack required
→ Openstack

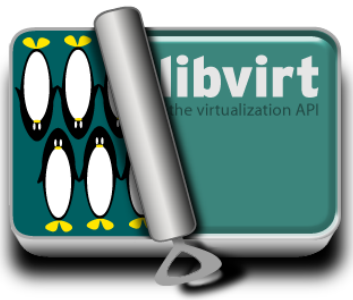
- already broadly used for private clouds
- good integration with KVM

Basic Architecture



- Preempt–RT as kernel
  - configuration and tuning according to https://rt.wiki.kernel.org
  - *rt thread throttling*
  - scheduling priorities: *kworker*, *ksoftirq*, *rcu*
  - interrupt affinities, power management
- *isolcpus*
  - for vCPUs
  - hyperthread siblings disabled
  - further isolated with `cpuset.cpu_exclusive`
- sufficient non–isolated CPUs
  - QEMU event threads
  - Linux base system, libvirtd, nova, neutron, ...

- only executes higher layers commands, no policy
- foundation Openstack builds on
- for RT–vCPUs all required controls upstream (>= 1.2.13)
  - CPU affinity setting for QEMU–threads (*cgroups/cpusets*)
  - scheduling parameter setting (policy, priority)
  - memory pinning (`mlockall()`)



Issues with CPU affinity setting

- *cgroup* operation ordering problems
- causes disturbance and starvation
- mostly solved in prototype
- WiP together with upstream

- several features already available
  - vCPU affinity setting
  - dedicated pCPUs
- RT–blueprint was available
  - introduced flavor property `hw:cpu_realtime`
    - memory pinning
    - vCPU scheduling policy and priority
  - implementation by Red Hat
  - similar implementation by Siemens

Deficits

- scheduling policy/priority were hard–coded
- second CPU mask required in nova, differentiate between RT and non–RT pCPUs for Openstack and other pCPUs

Corrected version merged

- about 2 weeks ago, implemented by Red Hat

PCI–Device assignment

- should be easy to do
  - libvirt and Openstack support it
- shortest possible way from guest to pNIC
- have to choose pNICs that the guest supports
- does not scale

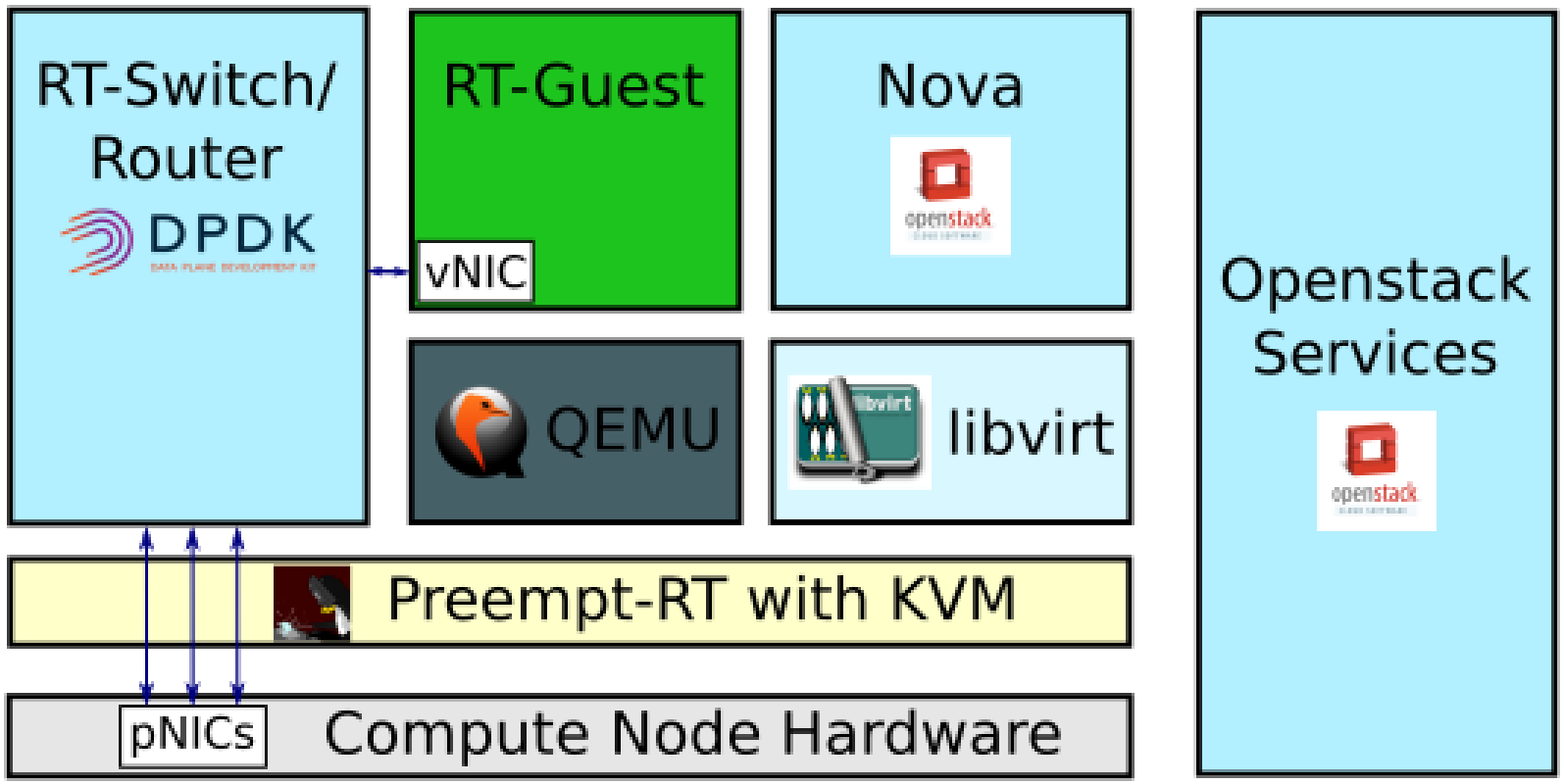
Issues

- Openstack lets you choose devices by device– and vendor–IDs
- have several NICs with matching IDs on different networks?
- problem is known upstream

Solution

- QEMU–wrapper script to rewrite arguments based on a name passed down by Openstack

Architecture with I/O



- get host kernel out of the loop
  - Ethernet, IP-stack, *–tables
  - device drivers
- implement RT–Switch with DPDK
  - `uio` device driver for all common NICs
  - Ethernet–stack
  - Real–Time scheduling parameters
  - small packet bursts and polling for low latency
- short way from guest to software–switch
  - shared ring memory with switch *vhost–user*
  - signaling via socket, but mostly polling

- requires another set of RT–pCPUs
  - can now be modeled in Nova
  - isolated against cgroup problems with `cpuset.cpu_exclusive`
- polling on high priority has potential to starve others
  - helps you find tuning problems, affinity bugs in libvirt, ...
- guests need *virtio* device driver

Issues

1. Openstack wants to fully manage networks (IPs, DNS, topology)
2. Openstack could not set memory backing store shared – might be solved upstream

Solution

1. implemented unmanaged network–class
2. QEMU–wrapper script to change arguments

Summary and Outlook

- Real–Time Cloud is feasible
- prototype available
- still requires good understanding of the stack
  - a lot was already there, still configuration is not trivial
  - found some issues on the way
    - partially fixed upstream
    - ongoing work
- open issues
  - how to properly protect your *isolcpus* from *cgroups*
  - *cgroups2* are coming soon . . .
  - RT–network integration with *neutron*, *Open vSwitch*
  - PCI–device assignment based on *bdf* ?
  - `share=on` for *virtio* memory backing store ?