libvirt integration and testing for enterprise KVM/ARM

Drew Jones, Eric Auger
Linaro Connect Budapest 2017 (BUD17)

Overview

- Enterprise guest requirements
- QEMU/KVM enterprise guest management
- libvirt
  - A one slide introduction
  - What’s new in libvirt for AArch64
  - libvirt and virt tool example uses
    - Describing a PCIe topology
    - Managing PCIe device assignment
    - Managing guest migration
- Virt stack verification
- Summary

Enterprise guest

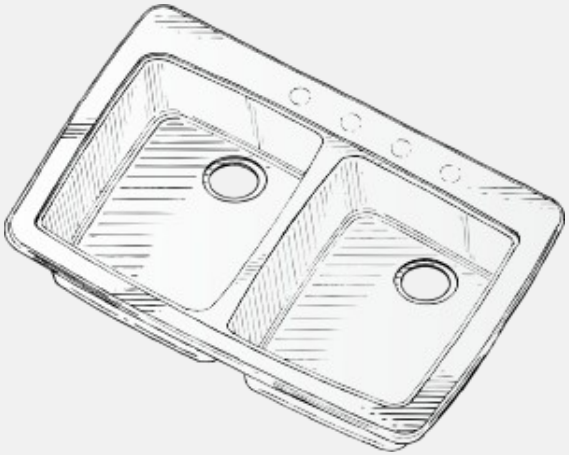
Features of an enterprise guest

- AArch64 machine features
  - Wide size range
    - memory, cores
    - disks, network adapters
  - PCIe device support
  - Firmware initiated boot
  - Standardized
- Virtual machine features
  - VirtIO
  - Generic devices
  - VM spec compliant

VirtIO

Additional guest requirements

- No unnecessary host exposure
  - Generic CPU model for vCPUs
- Dynamic memory sizing
  - Ballooning
  - ACPI hotplug events
- vCPU hotplug
  - ACPI hotplug events
- Device attach/detach support
  - SCSI and PCIe hotplug
- High performance support
  - PCIe device assignment – VFIO
  - Host pinning: vNUMA and virtual CPU topology



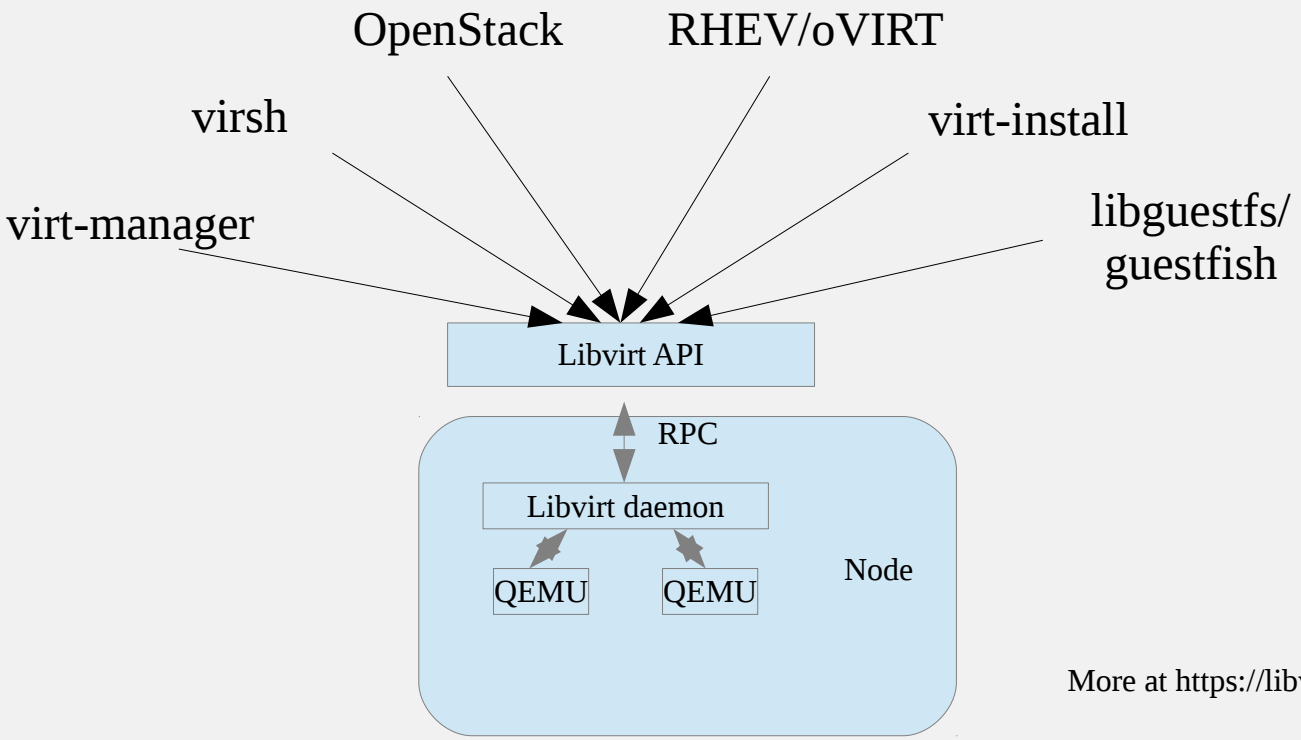
QEMU/KVM guest management

- QEMU interfaces are QEMU version dependent
- QMP access requires additional setup
- QEMU command line grows and grows…

```
qemu-system-aarch64 \
        -machine virt-2.9,accel=kvm,gic-version=3 \
        -cpu host,pmu=on -m 16384 -smp 48 \
        -drive file=/usr/share/AAVMF/AAVMF_CODE.fd,if=pflash,format=raw,unit=0,readonly=on \
        -drive file=/images/nvram/rhel_VARS.fd,if=pflash,format=raw,unit=1 \
        -display none -serial mon:stdio -no-user-config -nodefaults \
        -device ioh3420,port=0x8,chassis=1,id=pci.1,bus=pcie.0,multifunction=on,addr=0x1 \
        -device ioh3420,port=0x9,chassis=2,id=pci.2,bus=pcie.0,addr=0x1.0x1 \
        -device ioh3420,port=0xa,chassis=3,id=pci.3,bus=pcie.0,addr=0x1.0x2 \
        -device ioh3420,port=0xb,chassis=4,id=pci.4,bus=pcie.0,addr=0x1.0x3 \
        -device virtio-scsi-pci,id=scsi0,bus=pci.3,addr=0x0 \
        -drive file=/images/rhel.qcow2,format=qcow2,if=none,id=drive-scsi0-0-0-0,cache=none \
        -device scsi-hd,bus=scsi0.0,channel=0,scsi-id=0,lun=0,drive=drive-scsi0-0-0-0,id=scsi0-0-0-0,bootindex=1 \
        -netdev tap,id=hostnet0,script=no,downscript=no,ifname=tap0,vhost=on  \
        -device virtio-net-pci,netdev=hostnet0,id=net0,mac=52:54:00:62:c8:20,bus=pci.2,addr=0x0 \
        -device virtio-balloon-pci,id=balloon0,bus=pci.1,addr=0x0
```

libvirt

libvirt provides a consistent API



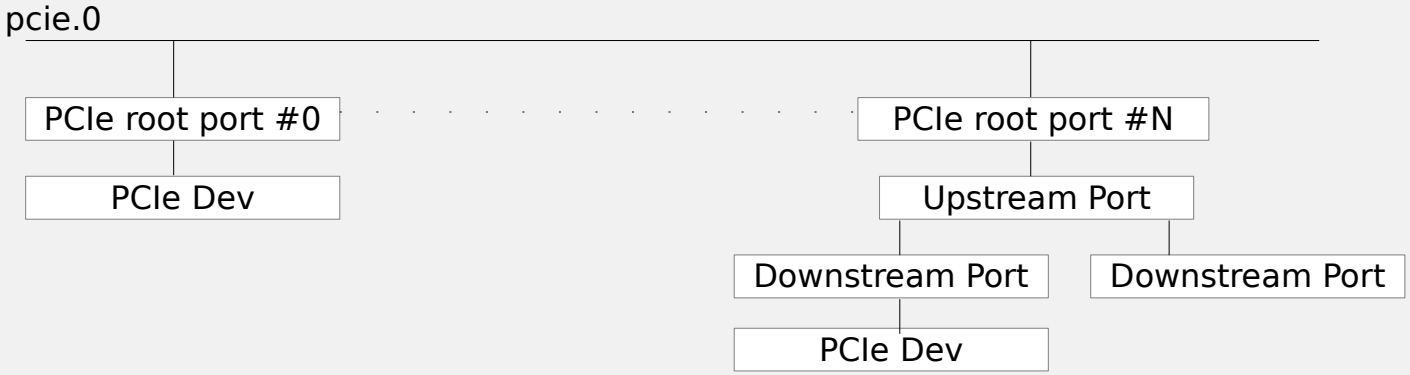
More at https://libvirt.org/apps.html

libvirt 3.x: new AArch64 support

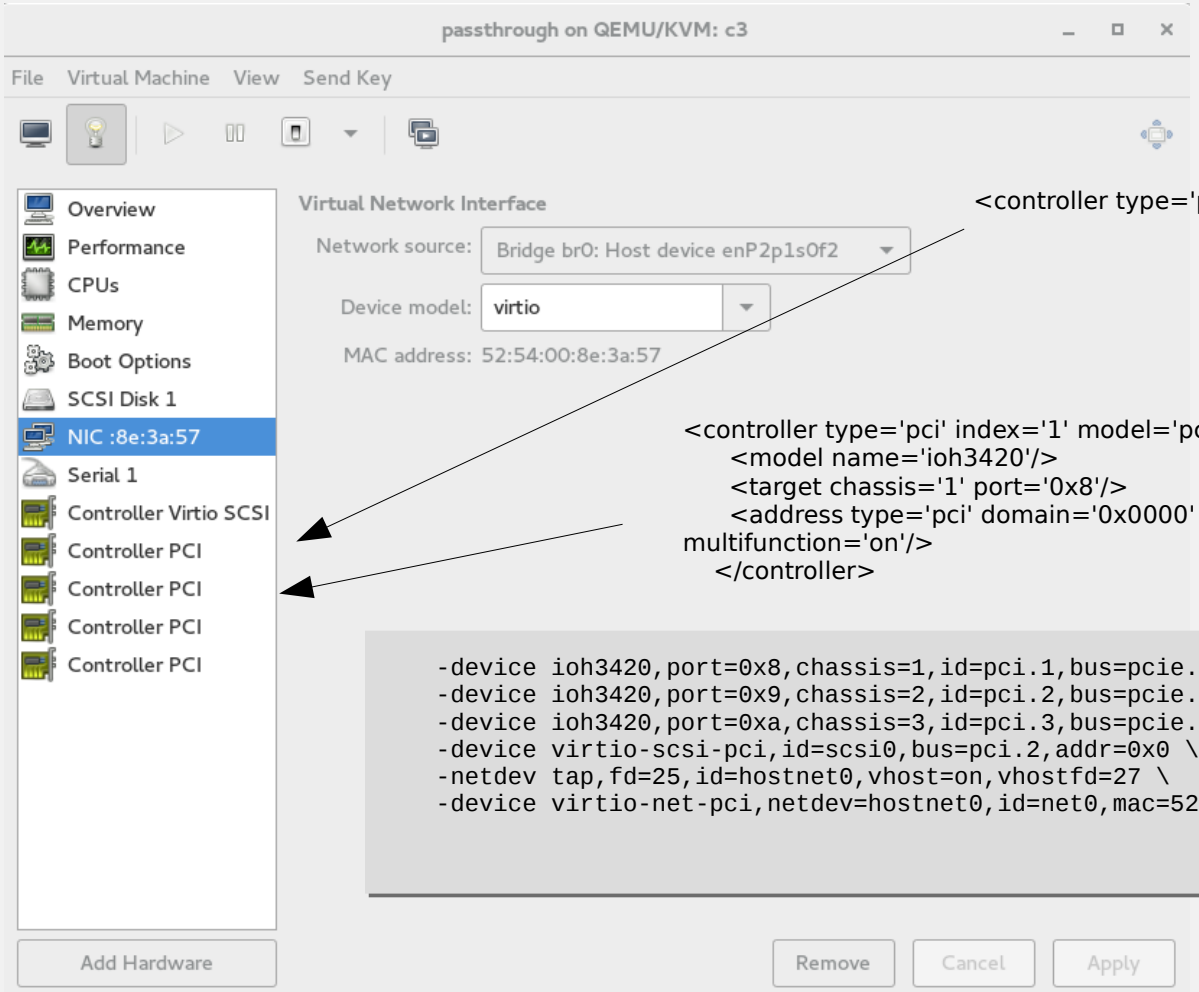
- New support
  - Use virtio-pci by default for mach-virt guests
  - PCIe placement
  - virtio-gpu
  - Auto GIC version selection
- Features for free (works like x86)
  - vPMU enablement
  - PCIe device assignment
  - Migration for GICv3/ITS

PCIe placement

- PCIe-only (no legacy PCI)
- Complies with QEMU docs/pcie.txt
- Root bus (pcie.0) and PCIe root ports at top
  - ioh3420 → generic root port
  - Flat hierarchy preferred (but PCIe switches are available)
- Root ports or switch downstream ports necessary for hotplug



PCIe placement: virt-manager




```
<controller type='pci' index='0' model='pcie-root'/>
```

```
<controller type='pci' index='1' model='pcie-root-port'>
    <model name='ioh3420'/>
    <target chassis='1' port='0x8'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x0'
multifunction='on'/>
  </controller>
```

```
-device ioh3420,port=0x8,chassis=1,id=pci.1,bus=pcie.0,multifunction=on,addr=0x1 \
-device ioh3420,port=0x9,chassis=2,id=pci.2,bus=pcie.0,addr=0x1.0x1 \
-device ioh3420,port=0xa,chassis=3,id=pci.3,bus=pcie.0,addr=0x1.0x2 \
-device virtio-scsi-pci,id=scsi0,bus=pci.2,addr=0x0 \
-netdev tap,fd=25,id=hostnet0,vhost=on,vhostfd=27 \
-device virtio-net-pci,netdev=hostnet0,id=net0,mac=52:54:00:8e:3a:57,bus=pci.1,addr=0x0
```

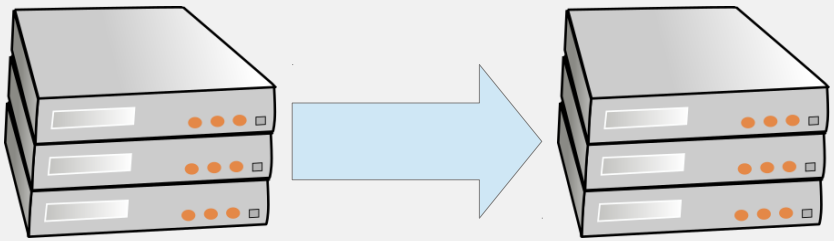
PCIe device assignment

- Minimum requirements: kernel 4.11, QEMU 2.8, libvirt 3.x
- virt-install: --host-device=pci_0005_90_00_0
- virsh: enumeration/dump, hot-plug/hot-unplug

```
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
     <address domain='5' bus='144' slot='0' function='0'/>
  </source>
</hostdev>
```

```
virsh nodedev-list --cap pci
virsh nodedev-detach pci_0005_90_00_0 --driver vfio
virsh nodedev-reattach pci_0005_90_00_0
virsh attach-device guest hostdev.xml
```

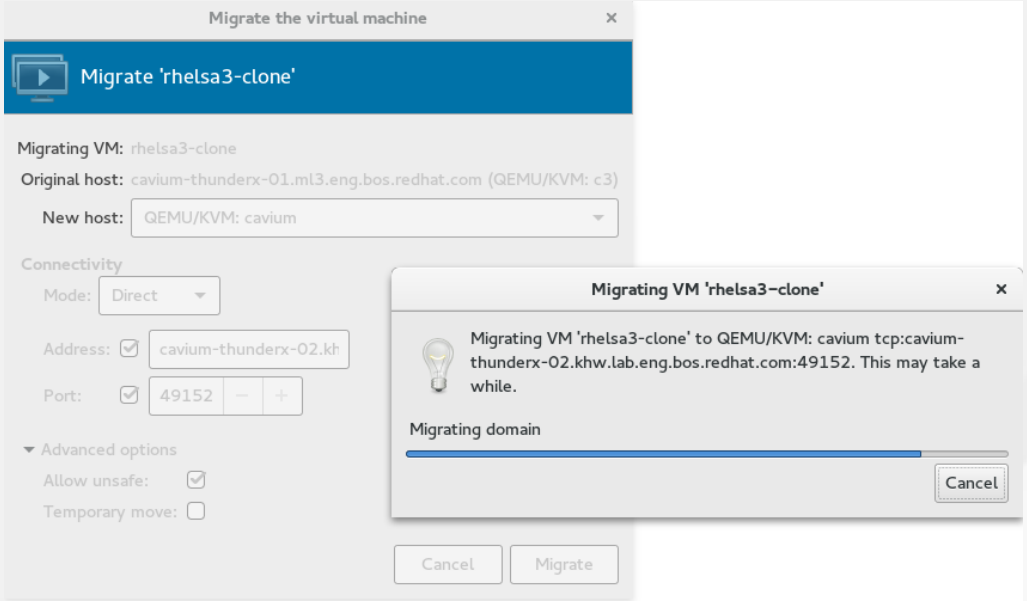
Migration



- GICv3 migration support
  - Minimum requirements: kernel 4.11, libvirt 3.x
  - QEMU: patches posted for 2.9
- ITS migration support
  - Patches posted
- Hosts need identical hardware
- Hosts do not need identical QEMU versions
  - mach-virt is versioned

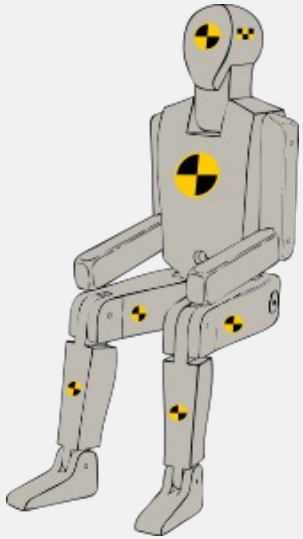
Migration with libvirt

- Tested with virsh and virt-manager
  - Used libvirt managed shared storage pool
  - virsh save / virsh restore
  - virsh migrate



Virt stack verification

avocado / avocado-vt



- avocado is a test framework inspired by autotest
- Supports plugins
  - avocado-vt is a plugin supporting QEMU/KVM/libvirt tests
- Can drive QEMU or libvirt / virt-tools
- Ported to AArch64
  - Teaching it defaults that match AArch64 enterprise guest setups
- Has thousands of test cases: functional and performance
  - E.g. guest install, boot, migrate
  - Not all tests applicable to a given target
  - Most libvirt based tests just work on AArch64
- Tests can be external testsuites (kvm-unit-tests)

Summary

Summary

• QEMU/KVM enterprise guests have complicated setups	libvirt provides common and stable APIs
• Many useful tools have been built on libvirt	Most libvirt-based tools just work when ported to AArch64
• Automated virt stack verification can target libvirt	avocado-vt
• Automated virt stack verification can benefit from libvirt	avocado-vt

References

- Libvirt and virt-tools
  - https://libvirt.org
  - https://www.virt-tools.org
- avocado-vt and kvm-unit-tests:
  - https://avocado-vt.readthedocs.io/en/latest/Introduction.html
  - https://www.linux-kvm.org/page/KVM-unit-tests

THANK YOU

kvm-unit-tests

- Simple guest kernel as a unit test
- Currently has support and tests for KVM’s emulation of
  - gicv2/v3
  - PMU
  - PSCI
  - Exits for QEMU PCI emulation
  - WIP: gicv3 ITS
  - WIP: nested virt
  - WIP: instruction emulation regression tests
- Gaining contributors
  - Contribute now and get a free baseball cap (*no, not really...*)