redhat.

The QEMU/KVM Hypervisor

Understanding what's powering your virtual machine

Dr. David Alan Gilbert
dgilbert@redhat.com
2015-10-14

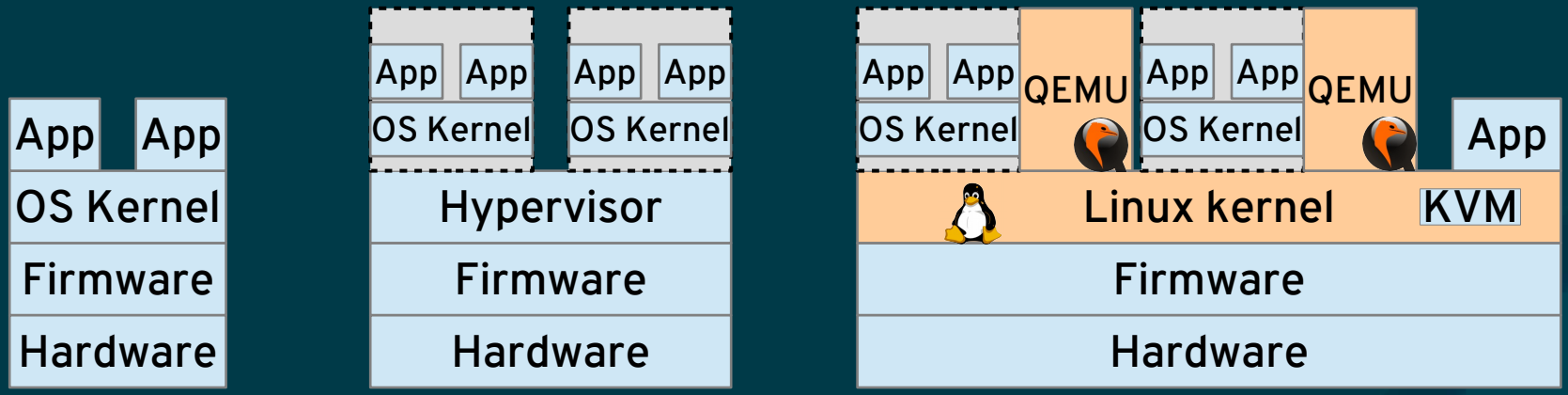
Topics

- Hypervisors and where QEMU/KVM sits
- Components of a virtual machine
- KVM
- QEMU
- Devices: Real, Emulated, and Paravirtualised
- Storage
- Networking
- Graphics
- Controlling QEMU
- Guest agents
- Migration tips
- Nesting

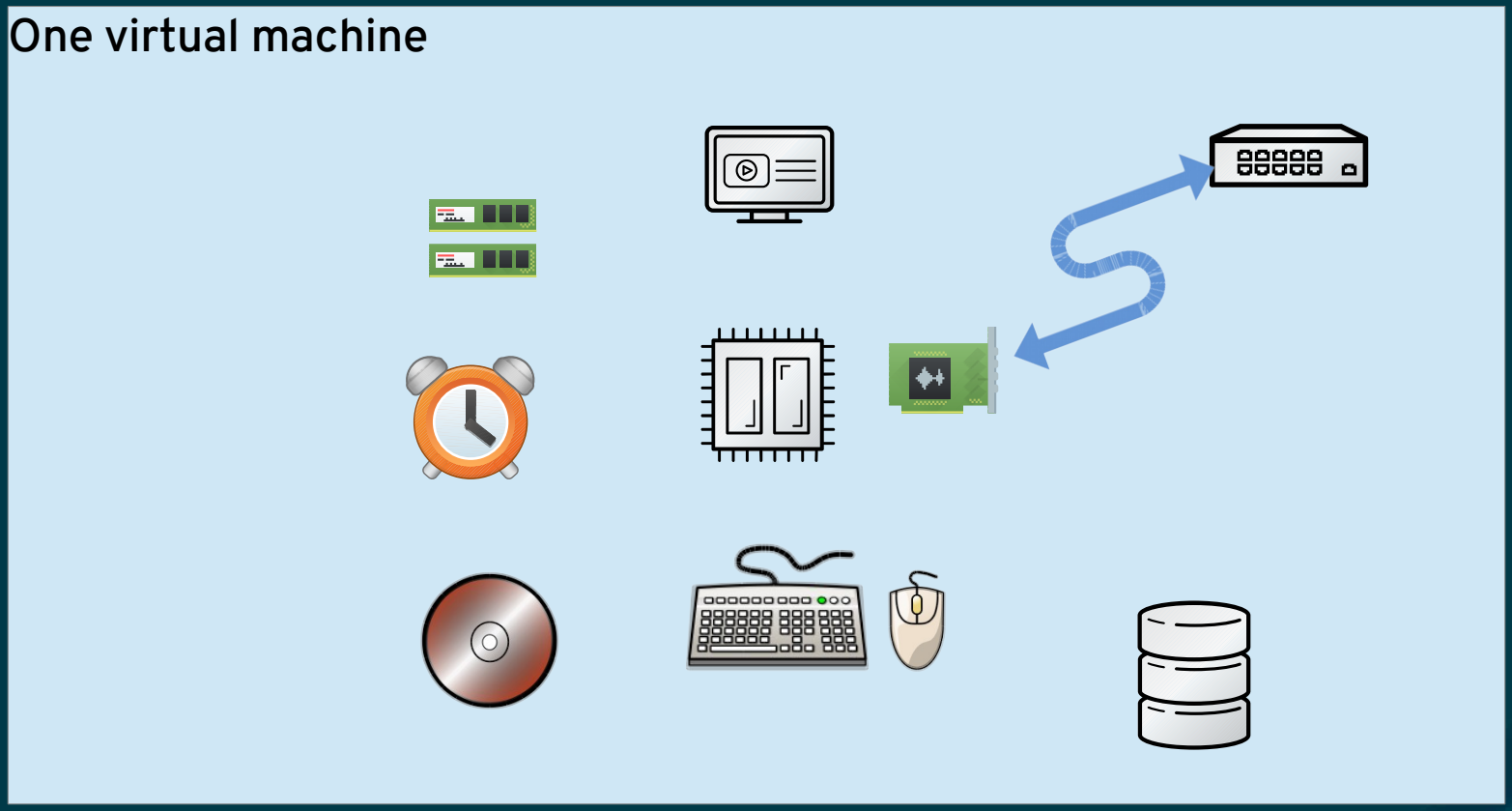
Hypervisors (aka Virtual Machine Manager)

& where QEMU/KVM fits

- Something that sits above a guest kernel
  - Supervisor old name for an OS Kernel
- Guest has it's own kernel
  - Can be a different OS
  - Host is isolated from guest kernel security issues



Components of a virtual machine



KVM

Linux kernel component for handling virtualisation

- How to virtualise a CPU – can't let the guest interfere with the host
  - Emulate? Slow
  - Emulate tricky instructions ? Messy, and still not that fast
  - Change the guest to remove tricky bits? Fast but limits the guests (Early Xen)
  - Use newer CPU features to do all the hard work
    - CPU put in a special mode then just runs the guests code natively
      - But exits back when it needs some help
    - KVM controls these features (Lots of architectures: x86, ARM, Power, mainframe etc)
    - Each vCPU appears as just a Linux thread
- Also has speedups for emulating some performance critical devices
  - Timers, networking

QEMU – Quick EMUlator

- Emulates CPU or uses KVM
  - Also used by Xen and other projects
- Emulates all the other devices
  - Disks, Networking, user input, graphics, timers, interrupt hardware etc
- Handles all the setup
  - Allocates memory, opens disk files
- Just a normal Linux process
  - Follows normal security rules (can add SELinux, cgroups etc to isolate)
- Provides control interfaces
  - Start/stop VM, dynamically add devices, debug output etc

Providing devices

- Emulate real devices
  - e.g. a specific model of IDE controller
  - + works with any OS that already knows the device
  - - Lots of legacy detail, can be slower, need to be bug compatible
- Paravirtual devices (e.g. virtio)
  - + optimised for virtual environment
    - No hardware oddities to match
  - - Needs special drivers in guest
- Real device pass through
  - Pass through PCI devices – very low overhead
    - Need IOMMU on host, can't migrate
  - Pass through USB
- Virtual function of real devices
  - One slice of a high end network card (SR-IOV), some GPUs

Storage

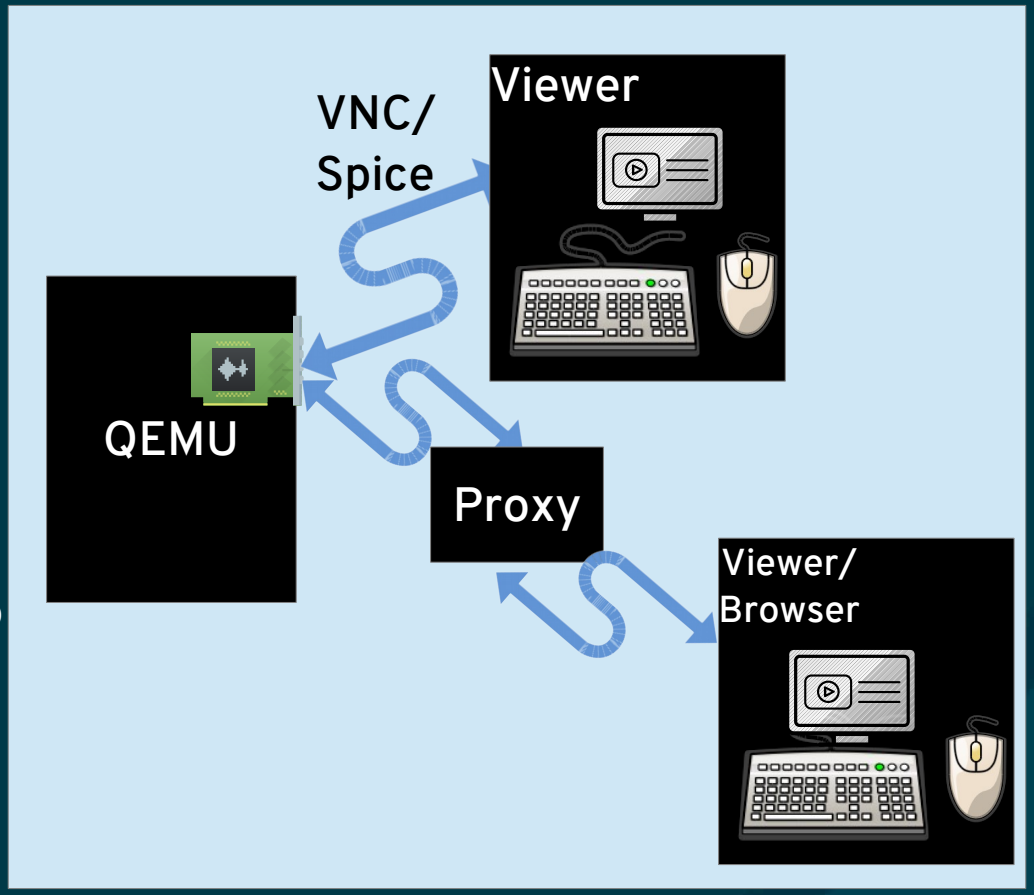
- Guest view:
  - IDE, SCSI, SATA, virtio-blk/scsi, USB storage
- Where?
  - Local file
  - Local block device
    - Take care of host security! - e.g. guest can label block device, or create LVM
  - Remote
    - Via iSCSI, HTTP, SSH, Gluster, CEPH, Sheepdog, NBD
    - Or via host kernel's remote iSCSI, NFS etc
- Format?
  - Raw
  - QCOW2
    - Snapshotting, COW, thin provision

Networking

- Guest view:
  - e1000, virtio-net (& some more obscure)
- QEMU passes packets to kernel
  - TUN/TAP setup, then connected to bridges/etc
  - Macvtap – can avoid the bridge setup (but can't connect to host)
  - QEMU userspace NAT
    - Bad idea except for simple testing/desktop use
- Vhost-net
  - Kernel extension to bypass QEMU
- Networking messy to setup by hand
  - Let the management layers do it!

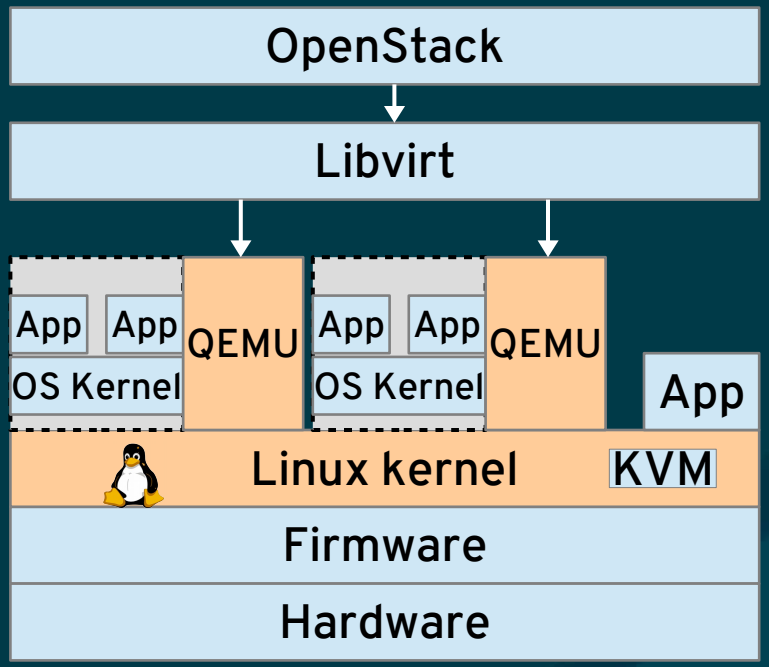
Graphics

- Guest view:
  - Cirrus, VGA, QXL/Spice
- Connection:
  - VNC, Spice, local GUI
- 3D acceleration being worked on
- Can pass spare host GPU through
  - But can be tricky, very model dependent
- Security:
  - Don't expose VNC, Spice direct to the internet, use a proxy



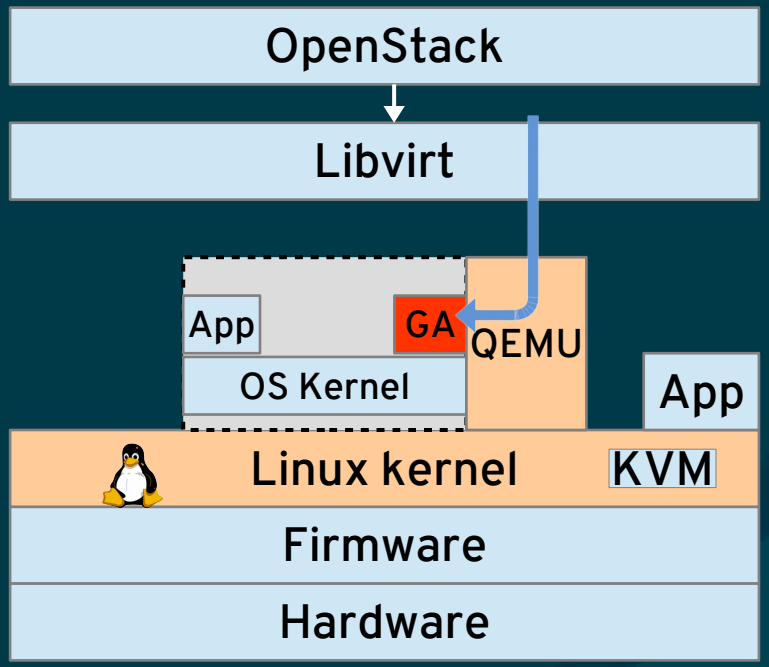
Controlling QEMU/KVM

- Huge command line
  - Specify each optional device (typically controller+device, e.g. IDE+file)
  - Specifies size of RAM, number of CPUs, type of CPU etc
  - Try and avoid driving it by hand – let *libvirt and higher levels do the hard work*
- QMP/HMP interfaces
  - HMP Human commands [mainly for debug]
  - QMP JSON commands
  - Runtime – e.g. hot add, status etc
  - Again let libvirt handle it
- Libvirt puts QEMU logs in /var/log/libvirt/qemu/guestname.log
- 'virsh' to control libvirt



Guest agents

- Optional installation inside the guest
  - As a package/application installed by the admin of the guest OS
- Various tools – e.g.
  - Set guest time
  - Suspend
  - Get stats
  - Read files

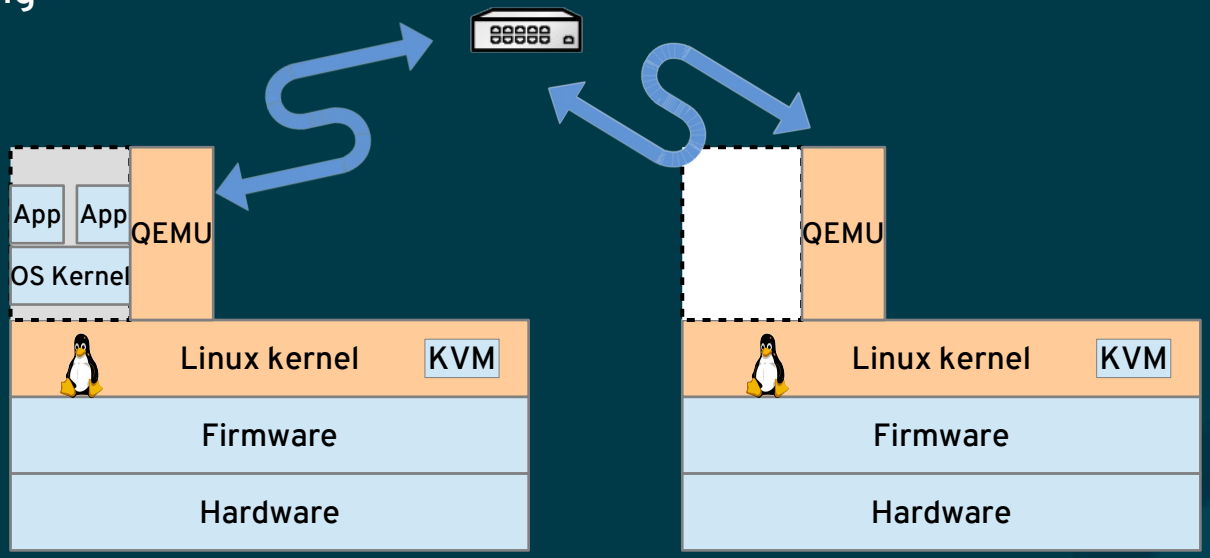


Migration

Move a VM from one host to another

- Non-Live
  - Pause guest, move it, carry on
- Live
  - Move it while it's changing
  - Needs fast network
- Networking gets reconfigured as it moves

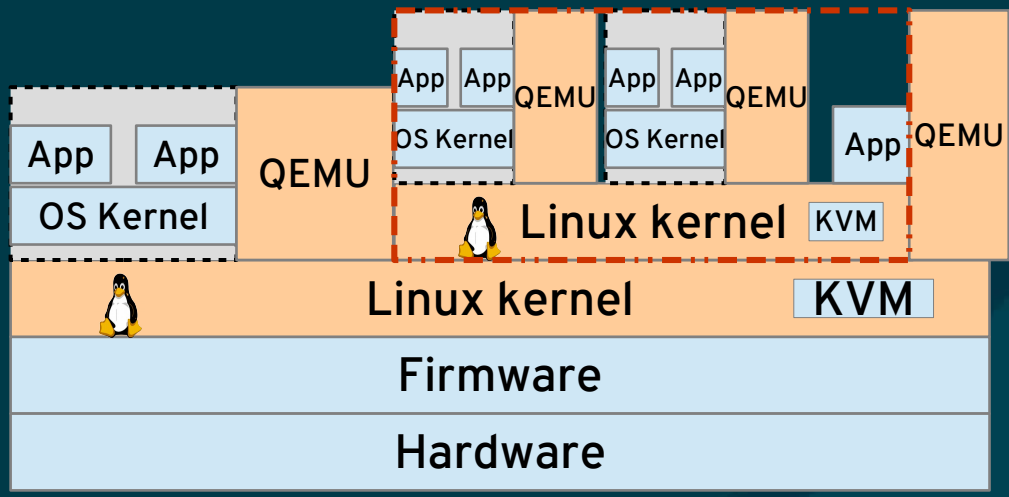
- Guest config must be IDENTICAL
  - Check machine type, devices, ROMs
- Guest CPU type
  - Pick one that is lowest common denominator in your cloud



Fun with Nesting

Very useful for development – probably not a good idea for production yet

- On most systems needs explicitly enabling in the top level (L0) host
- Can run an entire little cluster inside one host
- Convention for naming increases as you go down:
  - L0 – top level host
  - L1 – Guest of top level host
  - L2 – Guest of L1
- Networking can be 'interesting'
  - Make sure you don't use the same address range in L1 as L0
- Migrating a nest generally breaks
- Make sure KVM enabled in L1



redhat.

The QEMU/KVM Hypervisor

Understanding what's powering your virtual machine

Dr. David Alan Gilbert
dgilbert@redhat.com
2015-10-14