redhat.

Rethinking Machine Types

KVM Forum 2015

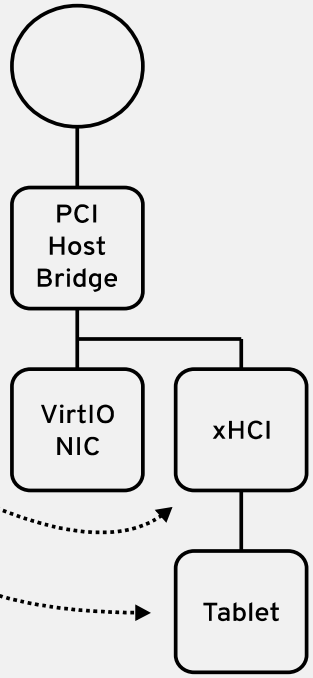
David Gibson
Senior Software Engineer, Virtualization
21 August 2015

What's the problem?

The qdev model

Command line options become virtual devices. Simple...

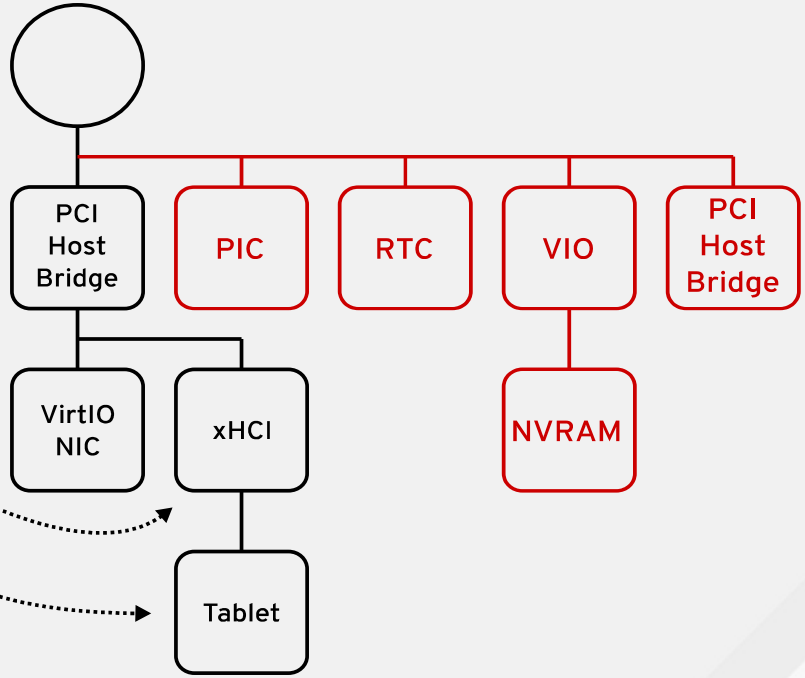
```
-nodefaults
-device spapr-pci-host-bridge,...
-device virtio-net-pci,...
-device nec-usb-xhci,...
-device usb-tablet,...
```



...and then there's machine type

Which adds a bunch of other stuff

```
-nodefaults
-device spapr-pci-host-bridge,...
-device virtio-net-pci,...
-device nec-usb-xhci,...
-device usb-tablet,...
-machine pseries
```



pc / q35 i386

- Legacy IO
- seabios
- i440FX / Q35
- ACPI
- ...

pseries ppc64

- PAPR hypercalls
- SLOF
- PAPR PCI Host
- PAPR VIO
- ...

virt aarch64

- GPEX PCI-E host
- Flash
- ARM GIC
- ...

xenpv i386

- Xen hypercalls
- ...

ppce500 ppc64

- Device Tree
- ePAPR boot
- ...

mac99 ppc

- Apple IO Hub
- Open Hackware
- Apple PCI Host
- ...

mac99 ppc

cubieboard arm

malta mips

The trouble with machine type

Machine type performs necessary system wide setup

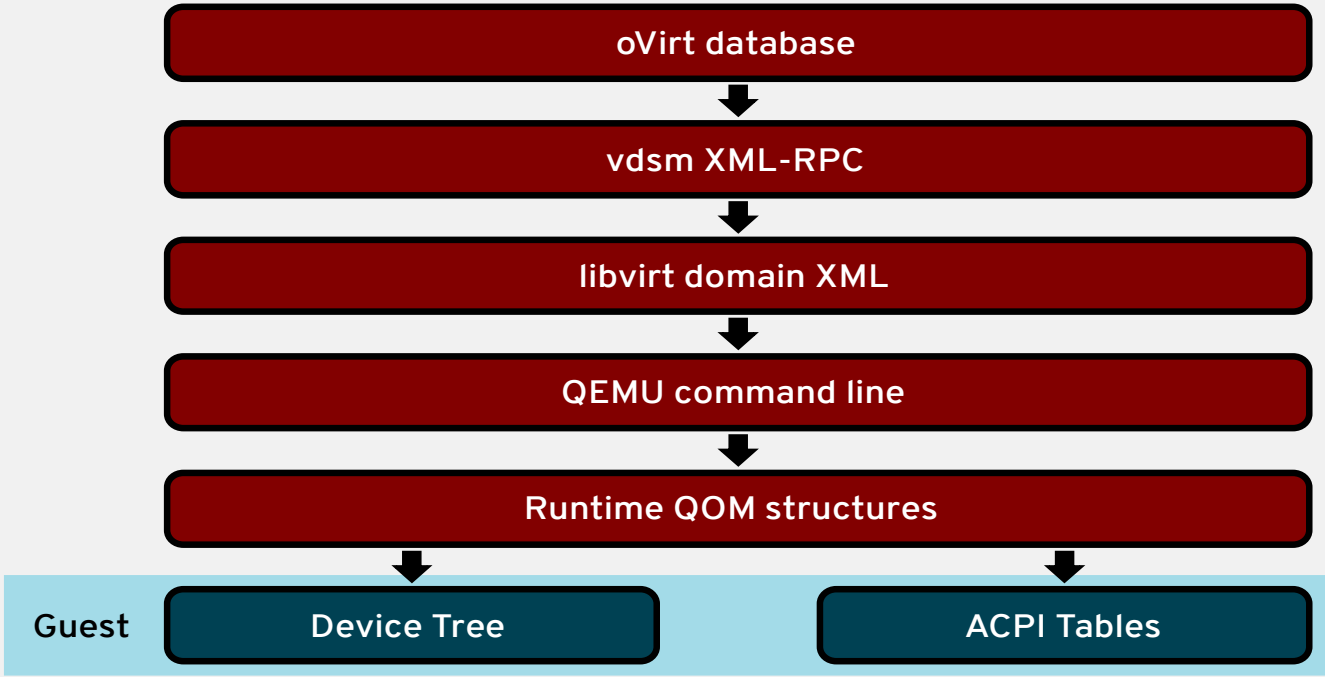
But it also..

- Adds “system” devices
  - Even with -nodefaults
- Behaviour can depend on machine options
- Or other options (-vga, -usb, -nographics)

PROBLEM #1
Machine type behaviour isn't easily discoverable

VM Hardware Description

How does the virtualization stack describe guest hardware?



PROBLEM #2

This many ways to describe virtual hardware? Really?

VM Hardware Description

Loose versus precise

20G storage

20G SATA disk,
on AHCI

20G SATA rev 2.0 disk,
on AHCI, rev 1.3,
in slot 2, function 1,
of PCI host bridge
at IO 0xabcd0000

VM Hardware Description

Loose versus precise (2)

- Humans and high-level tools want loose specification
  - ..except when they don't
- QEMU and guest need precise specification
- Converting loose → precise
  - Select default implementations
  - Add standard devices
  - Assign addresses

Migration

- Migration destination must have identical hardware to source
  - At least as far as the guest can tell..
- **Implementation** of the devices **can** change
  - Hosts with different paths to back-end storage
  - Host specific optimization hints
- libvirt manages migration
  - So it needs precise hardware information

PROBLEM #3
libvirt and qemu both have address assignment code

Hotplug

- QEMU must keep track of current hardware configuration
  - Including hotplugged (or unplugged) devices
- Must co-ordinate hotplug with guest
  - Platform specific protocols
- Combined with migration
  - Destination needs devices hotplugged on source
  - libvirt needs to track hotplugged devices

PROBLEM #4
libvirt and qemu track hotplugged devices in parallel

Recap

Problems with VM Hardware Description

PROBLEM #1
Machine type behaviour isn't easily discoverable

PROBLEM #2
This many ways to describe virtual hardware? Really?

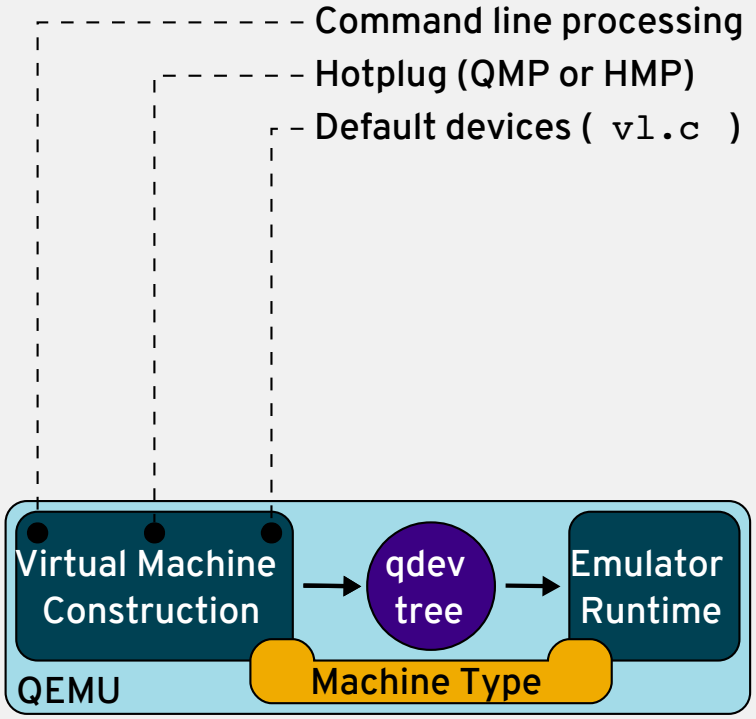
PROBLEM #3
libvirt and qemu both have address assignment code

PROBLEM #4
libvirt and qemu track hotplugged devices in parallel

How do we fix it?

QEMU

What needs to change?



Want a clear split between:

- Code building qdev tree
- Code using qdev tree

Pretty close already

- ..except for machine type

QEMU

Split machine type

MACHINE SCHEMA

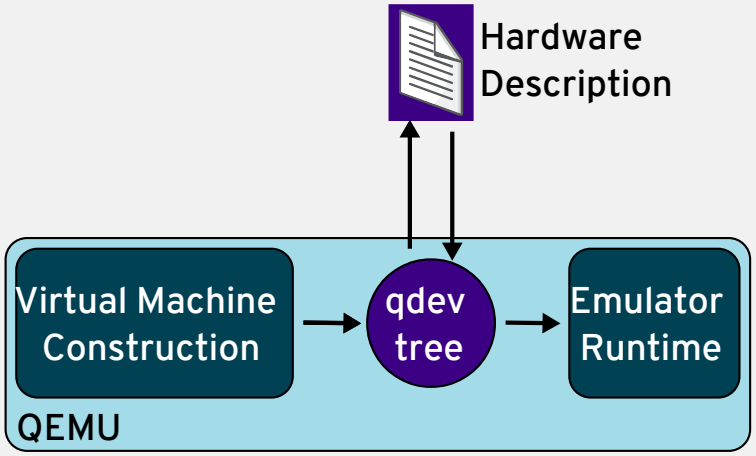
- Construct:
  - Platform essential devices
  - Platform default devices
    - (depending on options)
- Set up root bus
  - With class and parameters

ROOT BUS

- Subclass of SysBus
- Checks device dependencies
  - But doesn't try to fix
- Handles system wide reset
  - Firmware load / setup
  - CPU / memory initial state

QEMU

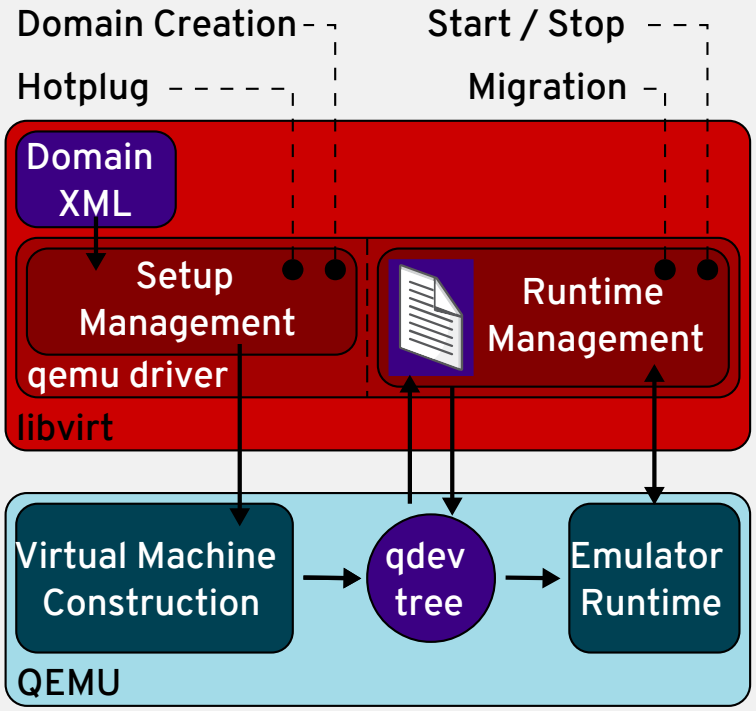
Expose hardware description



- Serialized hardware description format
  - Guest-visible & back-end pieces
- Allow hardware state to be extracted
  - Simply (no need to walk qtree)
  - Including hotplugged devices
- Allow specification to be re-inserted
  - Bypass machine construction
  - Bypass machine schema

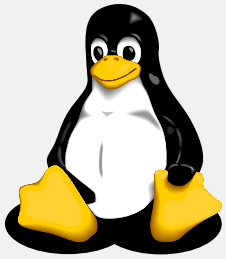
libvirt

HV drivers manage precise hardware description



- HV backends store precise description
- Creating new VM:
  - Translate XML into qemu options
  - Final VM description extracted
- (Re-)starting a VM:
  - Use stored precise description
- Can re-generate precise description
  - But requires guest restart
- Domain XML becomes loose only

The rest of the stack



Up the stack
Management tools

- Can keep using libvirt like now
- Optionally use new scheme
  - → detailed view of HW
  - → precise control of HW

Down the stack
Guest Operating Systems

- No change necessary
- Continue to use ACPI or DT
  - QEMU already creates this

Hardware Description Format

What would a consolidated format need?

- Tree structure
  - Express bus / bridge layout
- Extensible
  - Handle future hardware
- Separate guest visible and “back end” information
  - Work with one without parsing the other
- Preferably, already exists
  - Less to implement
  - Avoid N+1 standards

Hardware Description Format

What would a consolidated format need?

- Tree structure
  - Express bus / bridge layout
- Extensible
  - Handle future hardware
- Separate guest visible and “back end” information
  - Work with one without parsing the other
- Preferably, already exists
  - Less to implement
  - Avoid N+1 standards

Hardware Description Format

libvirt domain XML?

- XML heirarchy doesn't match bus heirarchy
- Guest and back-end info mixed
- Doesn't represent “system” devices
- Not clear from XML alone if it is a loose or precise description
- Needs XML parsing

libvirt XML is not well suited to precise hardware description

Hardware Description Format

Better ideas

Flattened Device Tree?

- Used by Linux guests
  - ppc, some ARM & MIPS
- Easy to parse, existing tools

- Lacks back end information
- qdev ↔ FDT may be complex
  - Some awkward redundancies

Linearize QOM?

- Easy to implement
  - QEMU already has JSON code
- Guest and back-end are separate

- Ties format to QEMU internals
- Might make future changes harder

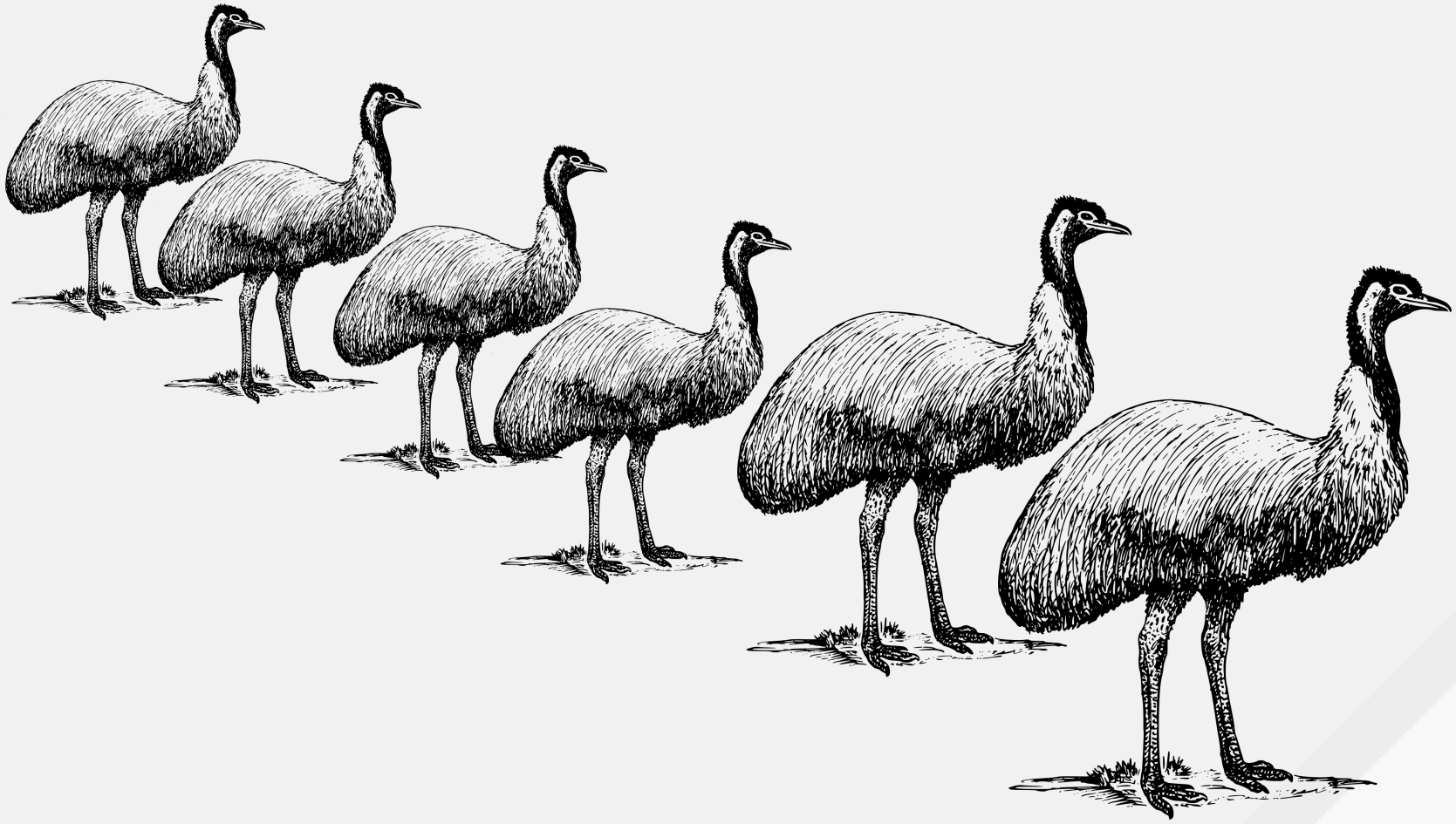
Getting started

How to get from here → there

1. Consensus amongst developers (QEMU and libvirt)
   - Is this a good approach?
   - Something like it?
2. Implement the machine type split
   - Has impacts across the tree
   - Enough people with enough time
3. Decide on a hardware description format
4. Implement import / export
5. Work outwards from there

Questions

http://people.redhat.com/~dgibson/kvm-forum-2015.pdf



redhat.

THANK YOU

plus.google.com/+RedHat

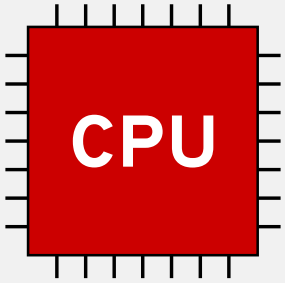
facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

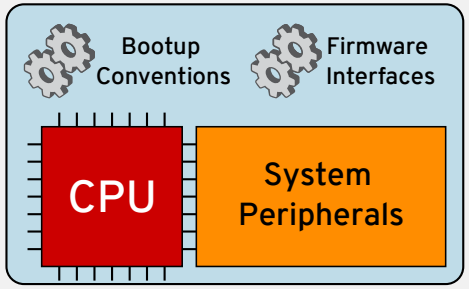
youtube.com/user/RedHatVideos

Architecture versus Machine Type



ARCHITECTURE
also known as

- CPU Architecture
- Instruction Set Architecture



MACHINE TYPE
also known as

- System architecture
- Sub-architecture
- Platform

Flattened Device Trees

Background

- Originated with Open Firmware (IEEE1275)
  - Conveys hardware information firmware → OS
  - Bus heirarchy tree
    - plus some special nodes
  - Device properties
    - key – value (bytestring) pairs
    - “Binding” documents
- Adapted to flattened form for use without full OF
  - Used by Linux for hardware discovery
    - All PowerPC and Microblaze
    - Some ARM, MIPS, and others