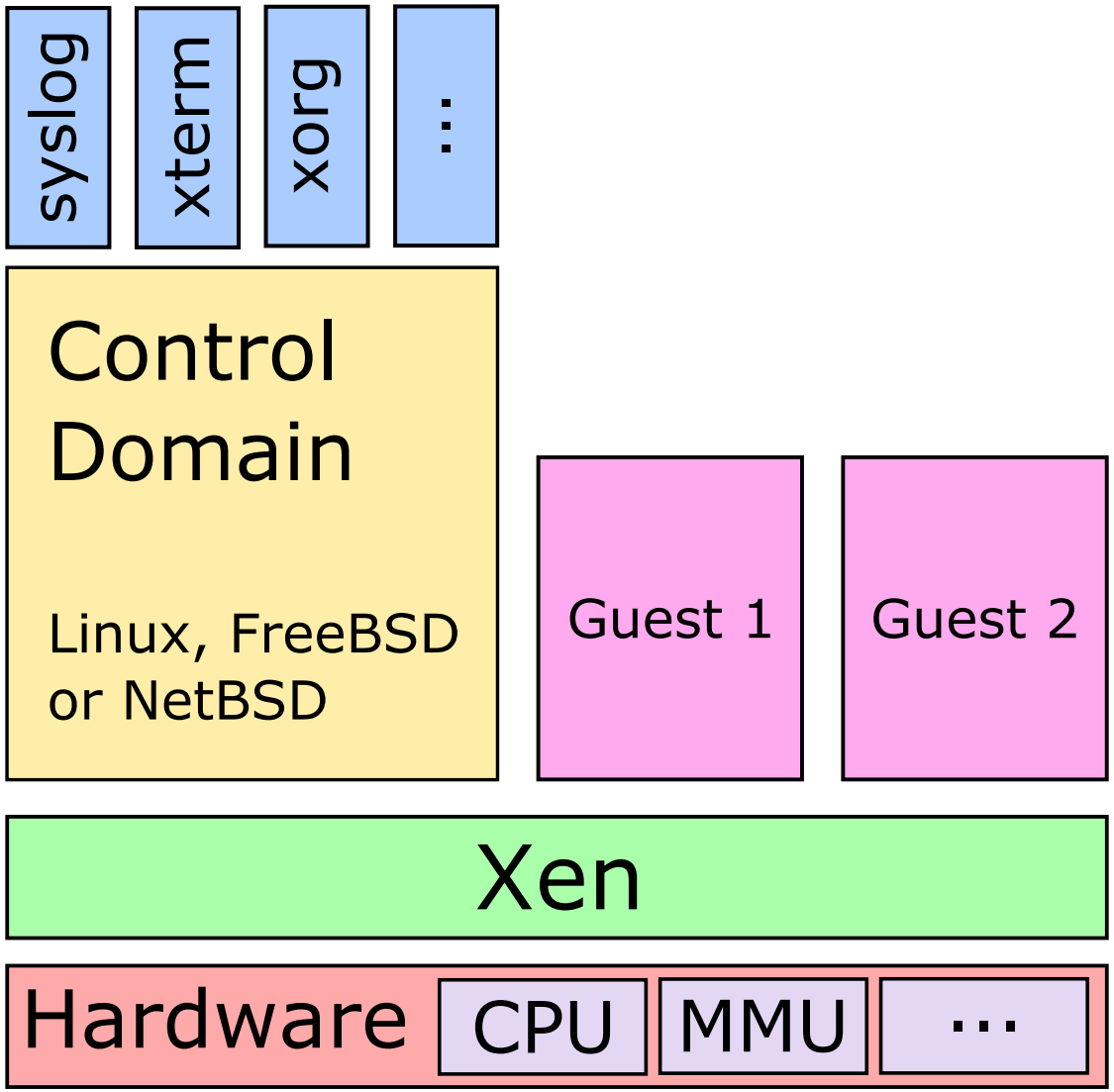
Towards a HVM-like Dom0 for Xen

Roger Pau Monné royger@FreeBSD.org

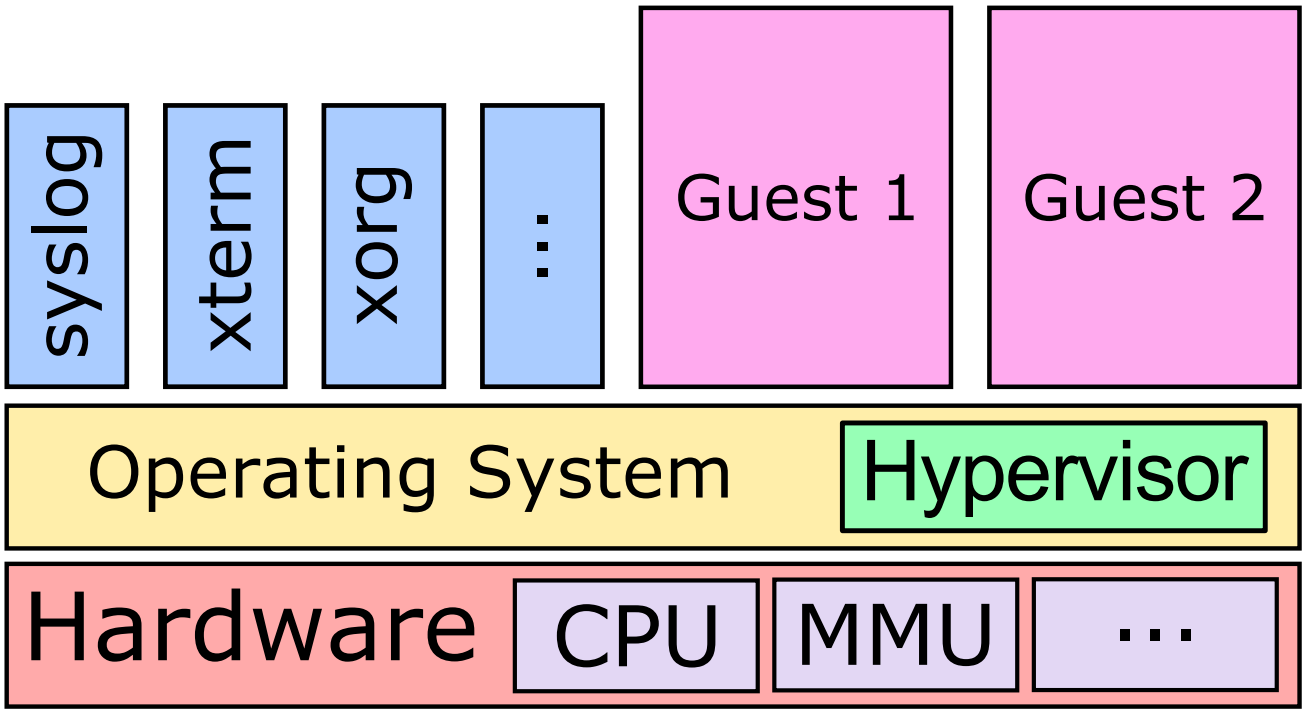
Tokyo – March 11th, 2017

Xen Project

Xen Architecture (type-1 hypervisor)



Type-2 hypervisor architecture



Current Dom0 interface

- Due to the nature of the Xen architecture, a different interface from the native one is used in order to perform several tasks:
  - MMU and privileged instructions.
  - CPU handling.
  - Setup and delivery of interrupts.
  - ACPI tables.

MMU and privileged instructions

- Traditional PV Dom0 uses the PV MMU:
  - Specific Xen MMU code in OSes.
  - Very intrusive.
  - Limited to 4KB pages.
  - Involves using hypercalls in order to setup page tables.
- Hypercalls are used in order to request the hypervisor to execute privileged instructions on behalf of the guest.

CPU handling

	Native	PV
Boot time enumeration	ACPI MADT	Hypercalls
AP bringup	Local/x2 APIC	Hypercalls
Hotplug	ACPI GPE and processor objects	Xenstore

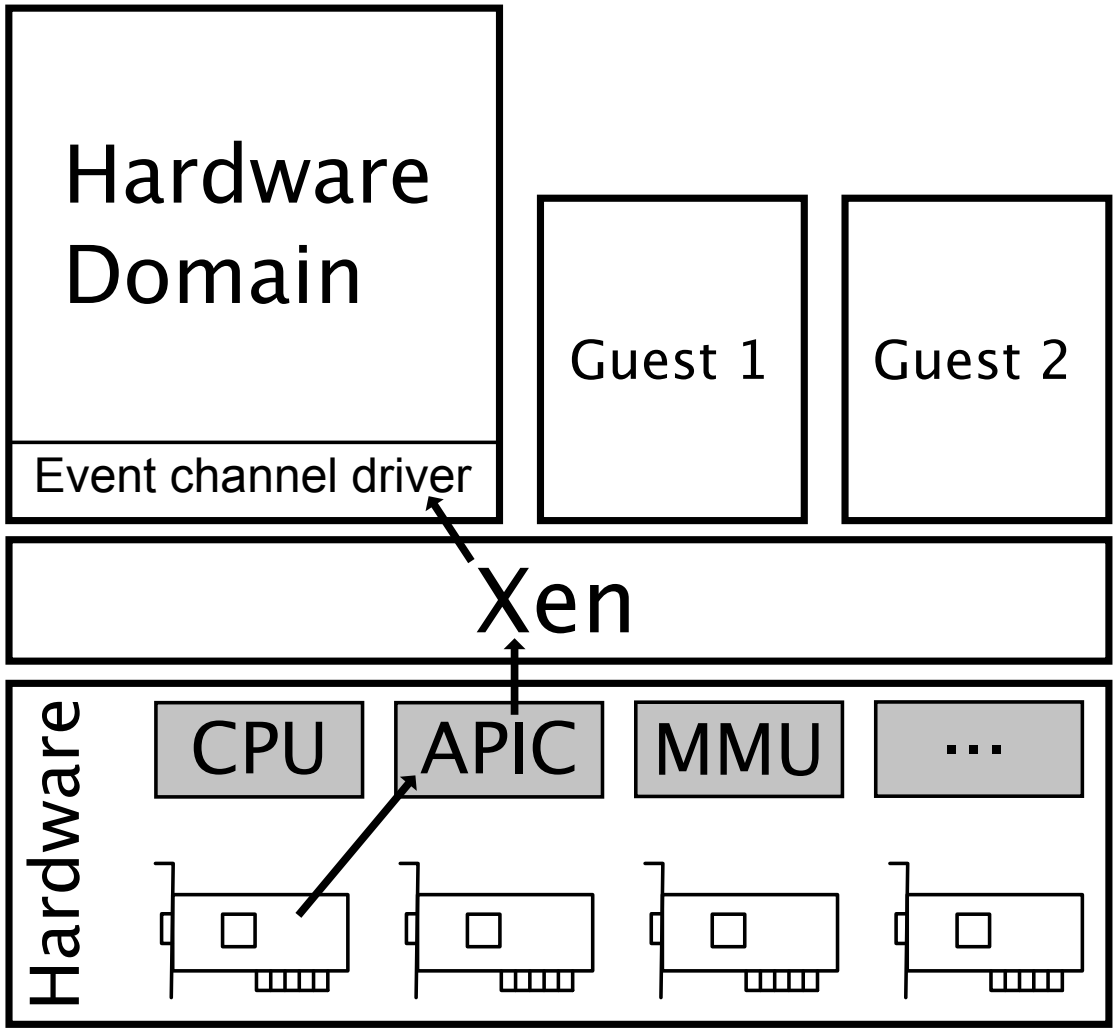
Setup and delivery of interrupts

- On x86 systems interrupts are delivered from the APIC to the CPU. There are several kinds of interrupts:
  - Legacy PCI: implemented using side-band signals, delivered to the IO APIC and then injected into the local APIC
  - MSI/MSI-X: implemented using in-band signals delivered directly to the local APIC.
- Configuration of interrupts is done from the PCI configuration space.

Setup and delivery of interrupts

- PV guests don't have an emulated APIC.
- Interrupts are delivered using event channels, the paravirtualized interrupt interface provided by Xen.
- Configuration of interrupts is performed using hypercalls.

Setup and delivery of interrupts



ACPI tables

- Two different kind of ACPI tables can be found as part of a system description:
  - Static tables: binary structure in memory that can be directly mapped into a C struct.
  - Dynamic tables: described using ACPI Machine Language (AML), an AML parser is required in order to access them. They can contain both data and methods.
- On a traditional PV Dom0 all tables are passed as-is to Dom0, and that forces Xen to use side-band methods for CPU enumeration.

ACPI tables

- Xen can only parse information from static ACPI tables.
- But there's information required by Xen that resides in dynamic tables:
  - Hotplug of physical CPUs.
  - CPU C states.
  - Sleep states.
- Dom0 has to provide this information to Xen.
- Although it would be possible for Xen to import a simple AML parser, there can only be one OSPM, so Xen could only look at the tables, but not execute any method.

A new interface for PVH Dom0

- As close as possible to the native interface.
- Only resort to hypercalls or similar options as last-resort.
- Take advantage of the hardware virtualization extensions.

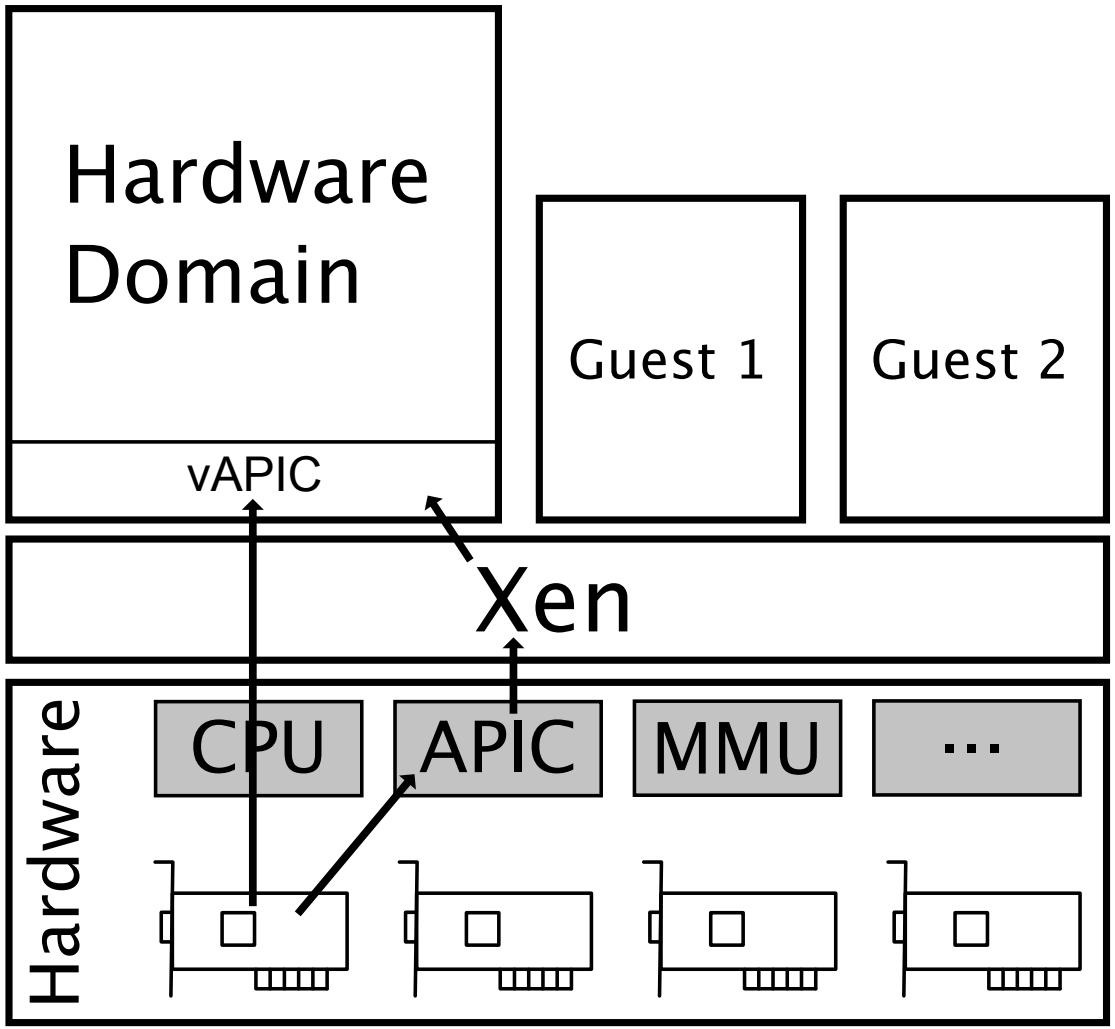
MMU

- Use the hardware virtualization extensions in order to provide a stage-2 page table for the guest:
  - Completely transparent from a guest point of view.
  - Guest can use the virtual MMU provided by the hardware.
  - Can use pages bigger than 4KB (2MB, 1GB).
  - No need for any modification of the OS.

Interrupt management

- Provide Dom0 with an emulated local APIC and IO APICs.
- Configuration of MSI/MSI-X interrupts from physical devices using the PCI configuration space.

Interrupt management



ACPI tables

- Provide Dom0 with the correct CPU topology in ACPI tables (MADT).
- Provide an extra SSDT table that contain processor objects for the Dom0 vCPUs[1].
- Hide native processor objects from Dom0 using the STAO.

[1] Still under discussion

How does this impact FreeBSD/Xen

- There is going to be some disruption in the FreeBSD/Xen port for Dom0.
- PVHv1 code is being removed from Xen, which is actively used by FreeBSD in order to run as Dom0.
- PVHv1 code will also be removed from FreeBSD probably at the same time (or close) to the addition of the PVHv2 code.
- PVHv2 is going to reduce the Xen-specific code in FreeBSD, a great deal of the code in sys/x86/xen/* will be removed.
- PVHv2 will automatically add support for running a FreeBSD/Xen Dom0 on AMD hardware.
- Less code to maintain, interface closer to bare-metal hardware.

Conclusions

- Introduce a new interface, try to reduce Xen-specific code in OSes.
- Take advantage of hardware virtualization extensions.
- Reduce the maintainership burden of OSes with Xen support.
- Simplify the Dom0 interface, in order to promote Xen support between OSes.

Q&A

Thanks

Questions?