Xen virtualization on FreeBSD

Roger Pau Monné

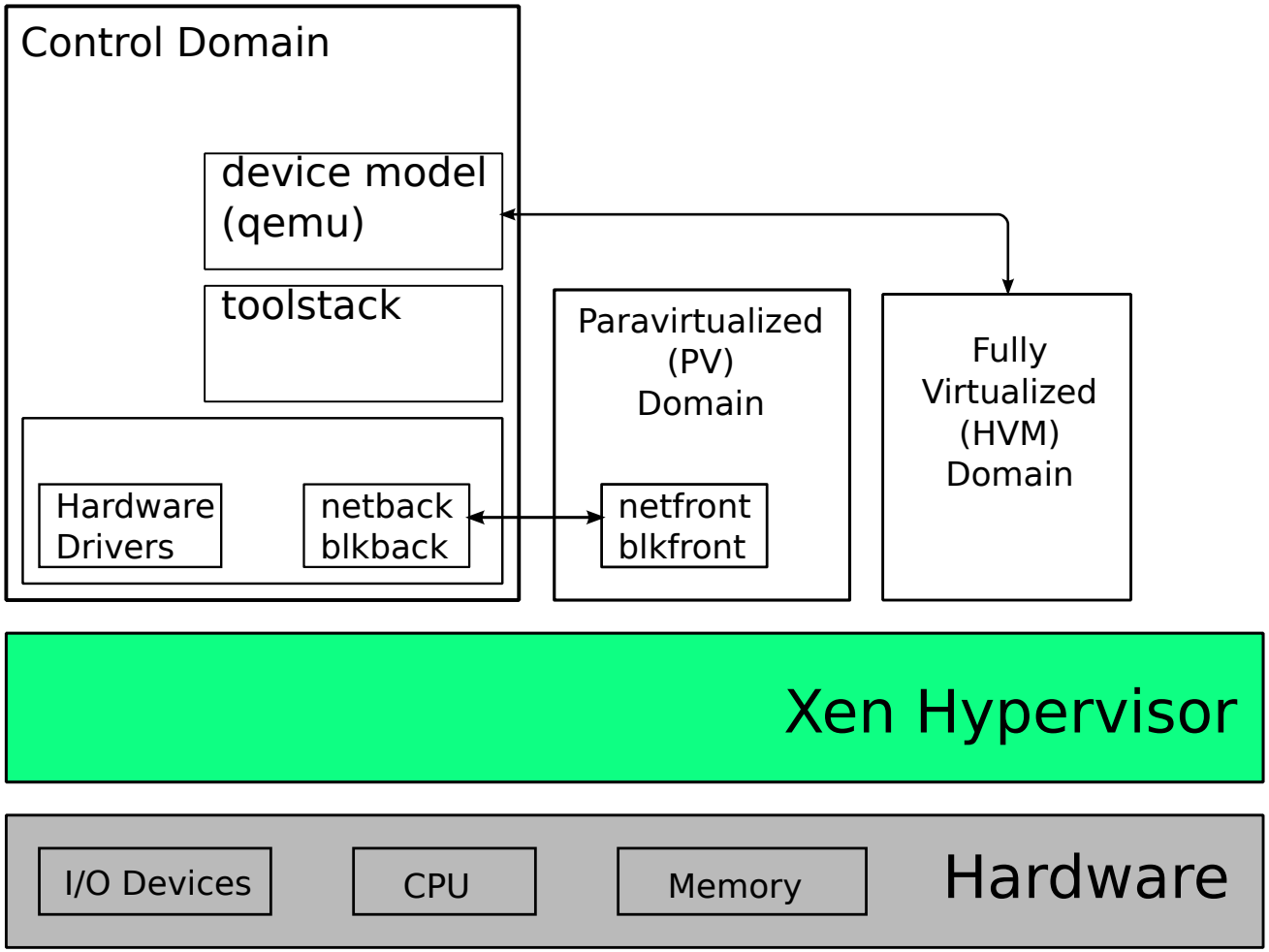
Tokyo – March 13, 2015

Xen Project

Goals of this presentation

- Description of Xen.
- Understanding how the Xen community works.
- A peek into Xen's new features.
- Recent work done in FreeBSD to improve Xen support.
- Introduction to the Xen toolstack.
- Hands-on session: setting up a FreeBSD/Xen Dom0.

Xen Architecture



Paravirtualization

- Virtualization technique developed in the late 90s.
- Designed by:
  - XenoServer research project at Cambridge University.
  - Intel.
  - Microsoft labs.
- x86 instructions behave differently in kernel or user mode, options for virtualization were full software emulation or binary translation.
  - Design a new interface for virtualization.
  - Allow guests to collaborate in virtualization.
  - Provide new interfaces for virtualized guests that allow to reduce the overhead of virtualization.
- The result of this work is what we know today as paravirtualiztion.

Paravirtualization

- All this changes lead to the following interfaces being paravirtualized:
  - Disk and network interfaces
  - Interrupts and timers
  - Boot directly in the mode the kernel wishes to run (32 or 64bits)
  - Page tables
  - Privileged instructions

Full virtualization

- With the introduction of hardware virtualization extensions Xen is able to run unmodified guests
- This requires emulated devices, which are handled by Qemu
- Makes use of nested page tables when available.
- Allows to use PV interfaces if guest has support for them.

The virtualization spectrum

VS	Software virtualization
VH	Hardware virtualization
PV	Paravirtualized

Poor performance

Room for improvement

Optimal performance

	Disk and network	Interrupts and timers	Emulated motherboard	Privileged instructions and page tables
HVM	VS	VS	VS	VH
HVM with PV drivers	PV	VS	VS	VH
PVHVM	PV	PV	VS	VH
PV	PV	PV	PV	PV

Xen community overview

- The Xen Hypervisor was released under the GPL2 on 2003.
- The Xen Project became a Linux Foundation Collaborative Project in 2013.
- Xen governance similar to the Linux kernel.
- Xen Project teams:
  - Xen Hypervisor.
  - ARM Hypervisor.
  - XAPI.
  - Mirage OS.
  - Linux PVOPS.

Xen governance

- Roles:
  - Maintainers: own one or more components in the Xen source tree.
  - Committers: maintainers that are allowed to commit changes into the source code repository.
  - Sub-projects and teams: run by individuals, projects are related or based on the Xen Project.
- See `http://www.xenproject.org/developers/governance.html` for more information.

Xen Hypervisor

- Main project, contains the hypervisor and the toolstack.
- Led by 5 committers; 2 from Citrix, 1 from Suse, 2 Independent.
- During the 4.5 release cycle the Xen Project had contributions from 93 individuals from 39 organizations, and 9 unaffiliated contributors.
- Organizations that contributed to the 4.5 release: Citrix, SUSE, Linaro, Verizon, Oracle, Intel, Amazon...
- Full list can be found at `http://wiki.xen.org/wiki/Xen_Project_4.5_Acknowledgements`.

Xen's new features

- Recent Xen changes:
  - Improved support for running Xen on ARM.
  - New virtualization mode: PVH.
  - As usual, improvements/bugfixes across all components.

Xen on ARM

- Started on 2011, focused on bringing Xen into ARM boards with virtualization extensions.
- Xen 4.5 is the recommended release for Xen on ARM.
- Has support for both 32 and 64bit ARM chips.
- More information can be found at `http://www.xenproject.org/developers/teams/arm-hypervisor.html`.

New x86 virtualization mode: PVH

- PV in an HVM container.
- PVH should use the best aspects from both PV and HVM:
  - No need for any emulation.
  - Has a "native" MMU from guest point of view.
  - Has access to the same protection levels as bare metal.
- Written by Mukesh Rathor @ Oracle.
- Significant revisions by George Dunlap @ Citrix.

The extended virtualization spectrum

VS	Software virtualization
VH	Hardware virtualization
PV	Paravirtualized

Poor performance

Room for improvement

Optimal performance

	Disk and network	Interrupts and timers	Emulated motherboard	Privileged instructions and page tables
HVM	VS	VS	VS	VH
HVM with PV drivers	PV	VS	VS	VH
PVHVM	PV	PV	VS	VH
PVH	PV	PV	PV	VH
PV	PV	PV	PV	PV

PVH technical overview

- Runs inside of an HVM container.
  - No PV MMU.
  - Runs with normal privilege levels.
- Disable HVM emulated devices.
- Uses PV start sequence.
  - Start with basic paging setup.
- Uses the PV path for several operations:
  - vCPU bringup.
  - PV hypercalls.
  - PV e820 memory map.
- Uses the PVHVM callback mechanism.

Differences with PV

- Pagetables controlled by guest.
- IDT controlled by guest.
- No pfn/mfn difference, guest only aware of gpfns.
- Native syscall/sysenter.
- No event/failsafe callbacks.
- Native IOPL.

Differences with PVHVM

- Requires Xen ELFNOTES in order to boot.
- Boots with paging enabled.
- Slight differences in the grant-table and xenstore setup.
- No emulated devices, so no emulated APIC or timers.

FreeBSD 9.x Xen support

- i386 PV port.
- HVM with PV drivers (both i386 and amd64).
  - Xenstore and grant-table implementations.
  - Event channel support.
  - PV Disk and Network front and backends.
  - Suspend and resume.

FreeBSD 10.x Xen support

- PVHVM.
  - Vector callback support.
  - Unified event channel code with the i386 PV port.
  - PV timer.
  - PV IPIs.
  - PV Suspend and resume.

FreeBSD PV timer

- Provides a singleshot event timer (et) implemented using VCPUOP_set_singleshot_timer.
- Provides a timecounter (tc) using the information provided by Xen in vcpu_time_info.
- Provides a clock using vcpu_time_info (that contains the uptime) and the wallclock time in shared_info.

FreeBSD PV IPIs

- On bare metal IPIs are handled/delivered via the local APIC.
- Can route those over event channels, since we can now deliver events to specific vCPUs.
- Removes the emulation overhead of using the LAPIC.

FreeBSD PV suspend/resume

- Rebind all IPI event channels.
- Rebind all VIRQ event channels (for the timer).
- Re-initialize the timer on each vCPU.
- Re-connect the frontends (disk, net).

Ongoing work in HEAD

- PVH DomU support.
- PVH Dom0 support.

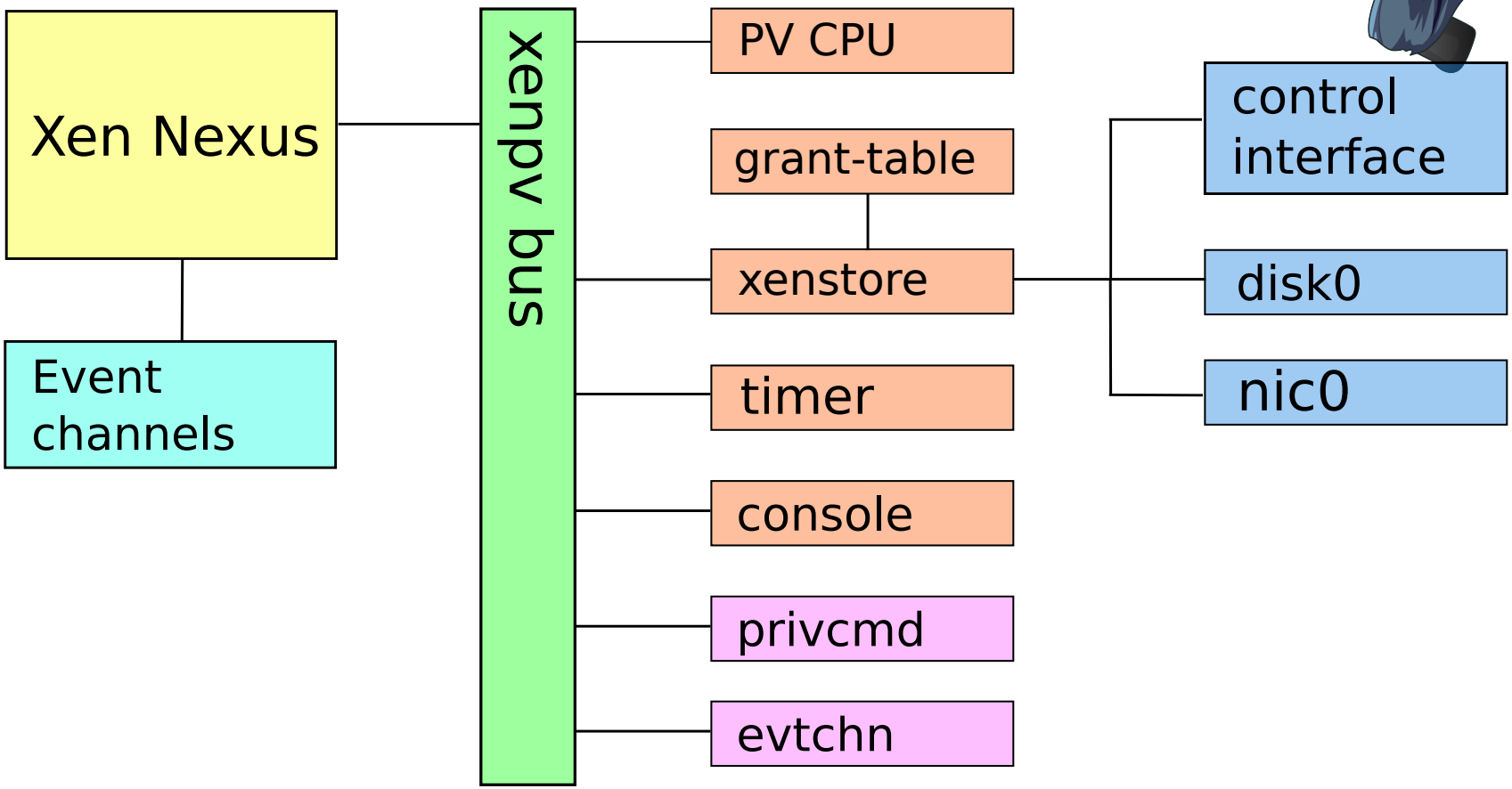
PVH DomU

- PV entry point into the kernel.
- Wire the PV entry point with the rest of the FreeBSD boot sequence.
- Fetch the e820 memory map from Xen.
- PV console.
- Get rid of the usage of any previously emulated devices (serial console, timers).
- PV vCPU bringup for APs.
- Hardware description comes from xenstore, not ACPI.

PVH Dom0

- Builds on top of DomU PVH support.
- Has access to physical hardware devices.
- Parses ACPI tables and notifies Xen about the underlying hardware.
- Special user-space devices are needed, so the toolstack can interact with Xen.

Architecture overview



Dom0 user-space devices

- privcmd:
  - Allows the toolstack to perform hypercalls.
  - Allows mapping memory from foreign domains.
- evtchn:
  - Allows registering event channels from user-space applications.
  - Allows receiving and sending event channel interrupts.

Xen toolstack

- Xen used to have two different toolstacks: xm and xl.
- xm deprecated for serveral releases, finally removed in Xen 4.5.
- xl is built on top of libxl (libxenlight), a library to interact with the hypervisor.
- libxl features:
  - libxl provides a stable API.
  - Coded in C (xm was built on python).
  - Small and efficient code-base.
  - libvirt driver built on top of libxl.

xl

- The default toolstack to interact with Xen is xl.
- xl is a cli utility.
- Configurations for VMs stored as plain text files.
- xl provides a set of commands to manage the hypervisor.
- Doesn't do any kind of storage/network management.
- Users that want a more advanced toolstack should use libvirt/CloudStack/OpenStack...

Example xl configuration file

```
kernel = "/root/vmlinuz-3.14.0"
ramdisk = "/root/initrd.img-3.14.0"
extra="root=/dev/xvda1"

vcpus = 4
memory = 2048

name = "test"

vif=[
    'bridge=bridge0,mac=00:16:3e:48:e2:a8'
]

disk=[
    '/root/test.img,raw,xvda,rw'
]
```

Pending work items

- Improve robustness and compatibility of if_xn/xnb (PV nic).
- Add some additional user-space devices to interact with Xen:
  - gntdev: allows user-space applications to map grants.
  - gntalloc: allows user-space applications to share memory using grants.
- Add a FreeBSD Dom0 to the Xen automatic test system (OSSTest).
- Test on different hardware.

Conclusions

- FreeBSD/Xen support is evolving from HVM → PVHVM → PVH.
- Initial FreeBSD PVH Dom0 support committed to HEAD.
- Using Xen allows to provide a fully featured virtualization platform based on FreeBSD.

Q&A

Thanks

Questions?