How we ported FreeBSD to PVH

A description of PVH and how to port an OS to it

Roger Pau Monné

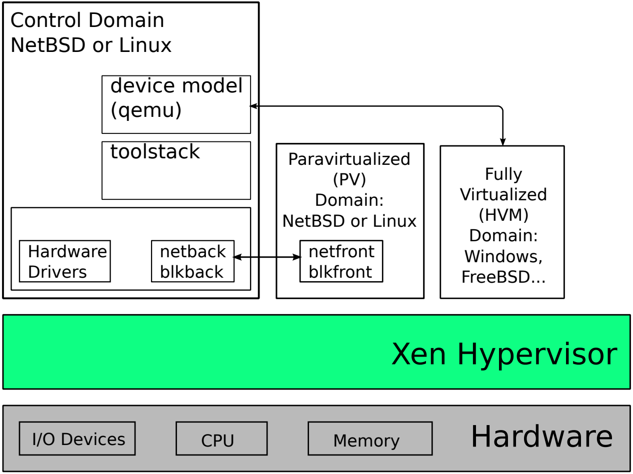
Brussels – February 2, 2014

Xen Project

Goals of this presentation

- Understand the motivation for PVH.
- Overview of the PVH architecture.
- Provide hints and implementation details about how to implement PVH support in an OS.

Xen Architecture



Issues with PV

- Pure PV guests require a PV MMU.
- Performance of 64bit guests syscalls.

Issues with HVM

- Requires one Qemu per guest.
- Legacy boot.
- Emulated devices in Xen.

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 virtualization spectrum

VS	Software virtualization
VH	Hardware virtualization
PV	Paravirtualized

- Red: Poor performance
- Yellow: Room for improvement
- Green: 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.
- gpfn in pagetables.
- IDT controlled by guest.
- No pfn/mfn difference, guest only aware of gpfns.
- Native syscall/sysenter.
- No event/failsafe callbacks.
- Native IOPL.

Differences with HVM

- 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.

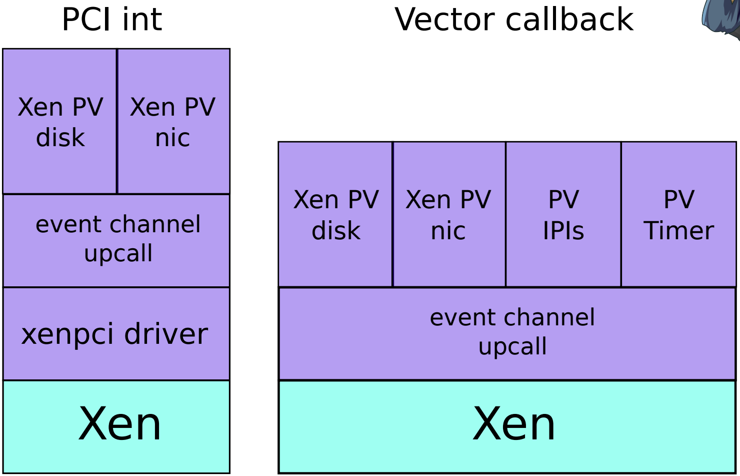
Not yet working on PVH

- Support for AMD hardware.
- 32bit guests.
- vtsc.
- Shadow mode.
- vCPU hotplug.
- Migration.

FreeBSD status before PVH

- Full PVHVM support (minus interrupt remapping).
  - Xenstore and grant-table implementations.
  - Event channel support.
  - Disk and Network front and backends.
  - Vector callback.
  - PV timer.
  - PV IPIs.
  - PV suspend/resume.

FreeBSD event channel callback handling



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.

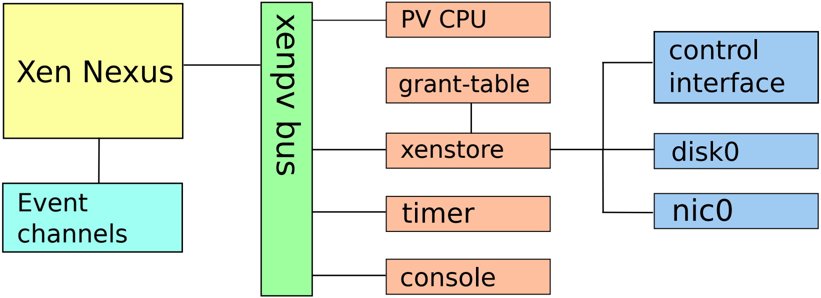
FreeBSD PV suspend/resume

- 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).

What's missing?

- 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.

The resulting architecture



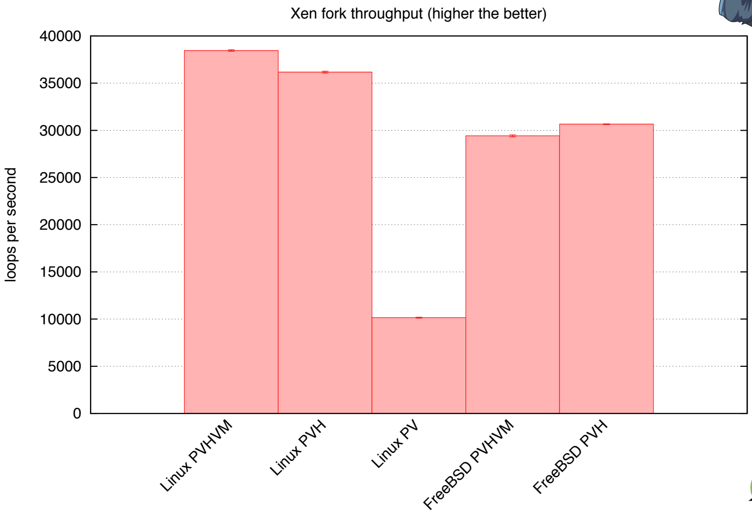
The benchmark host

- Dell Precision T3500.
- 8-way Intel(R) Xeon(R) W3550 @ 3.07GHz
- 6GB of RAM.
- Storage on LVM.
- dom0_max_vcpus=1 dom0_mem=1024M
- Xen 4.4-rc1
- Linux 3.12.0-rc7

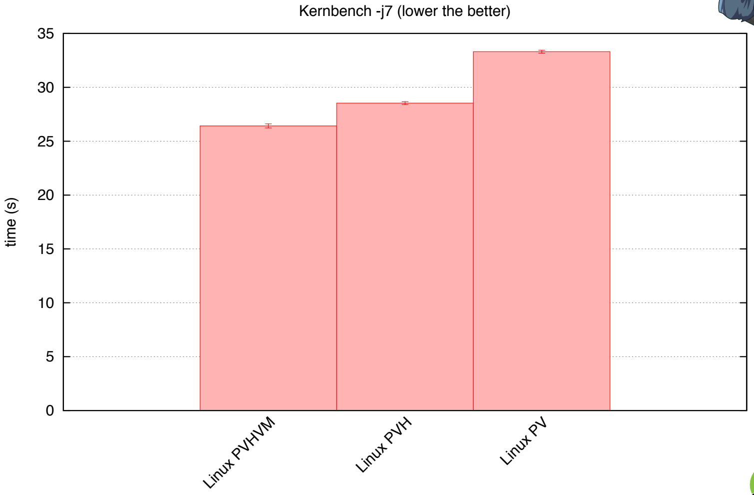
The benchmark guests

- 7 vCPUs.
- 4GB.
- 1 nic, 1 disk.
- Same VM used for all the tests, only difference is the mode in which the guest runs.
- PVHVM tests using upstream Qemu as shipped by Xen.
- Linux using git://git.kernel.org/pub/scm/linux/kernel/git/konrad/xen.git branch devel/pvh.v13
- FreeBSD using git://xenbits.xen.org/people/royger/freebsd.git branch pvh_v10

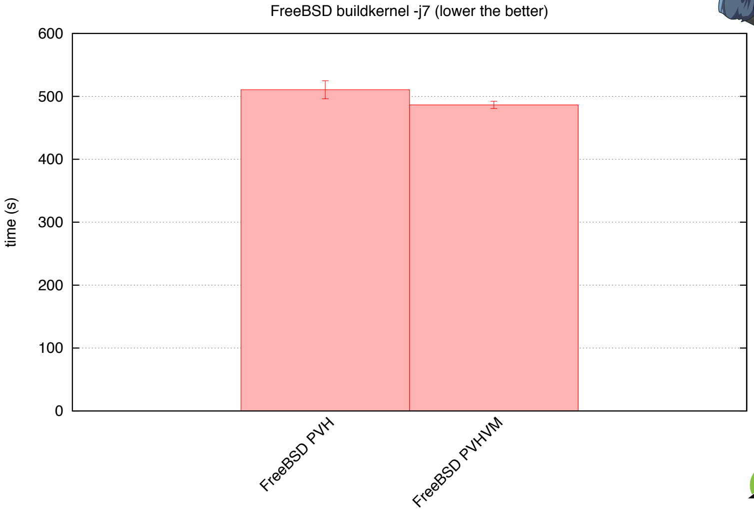
UnixBench fork benchmark



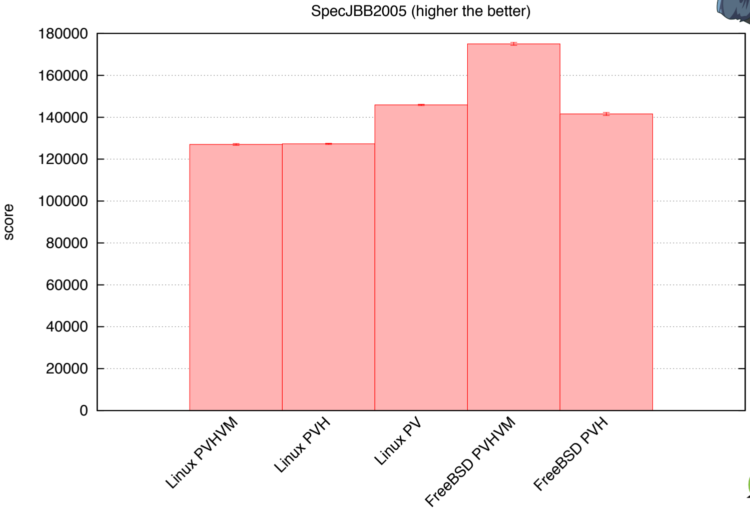
Kernbench


Kernbench -j7 (lower the better)
time (s)
0
5
10
15
20
25
30
35
Linux PVHVM
Linux PVH
Linux PV

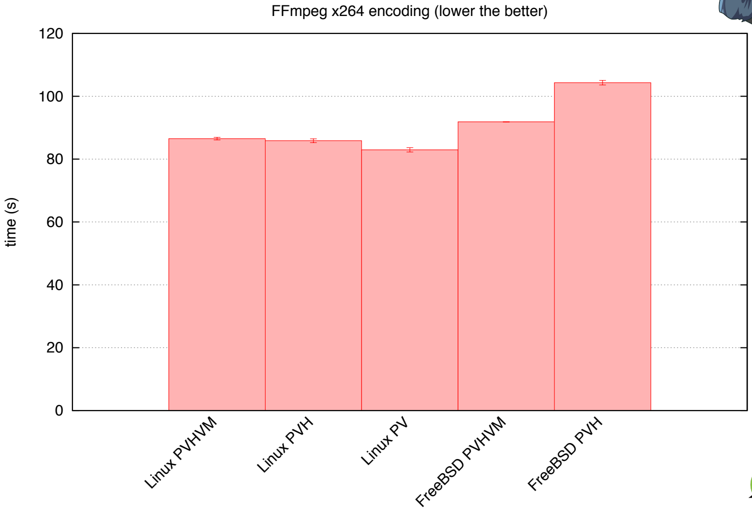
Buildkernel



SpecJBB2005



FFmpeg encoding with 12 threads



Future work

- FreeBSD side:
  - Full FreeBSD PVH Dom0 support on the kernel side.
  - Port the Xen toolstack (libxc, libxl, Qemu...) to FreeBSD.
- Xen side:
  - AMD support.
  - 32bit guests.
  - Migration.
  - PCI pass-through.

Conclusions

- PVH should provide a performance similar to PVHVM while being more "lightweight".
- Not as intrusive as pure PV for guest OSes.
- PVH shares a lot of similarities with PVHVM.
- Provides a nice transition path HVM → PVHVM → PVH.
- Can make use of new features in hardware assisted virtualization.

Goals

- Understand the motivation for PVH.
- Overview of the PVH architecture.
- Provide hints and implementation details about how to implement PVH support in an OS.
- Interest people in collaborating on the ongoing PVH work.

Q&A

Thanks
Questions?