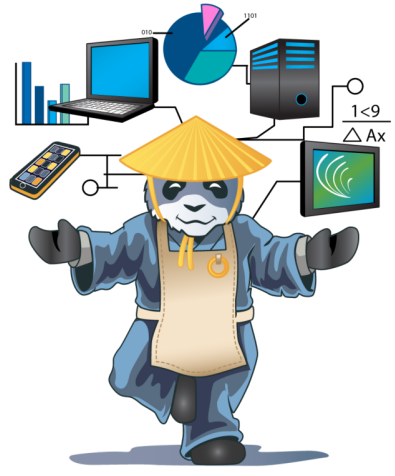
How to port your BSD to run as Xen on ARM guest

Julien Grall
julien.grall@citrix.com

BSDCan 2015

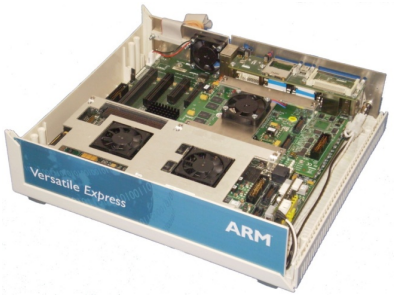


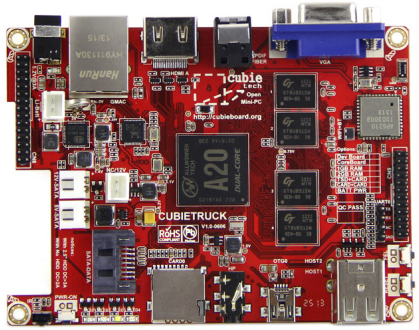
Xen Project

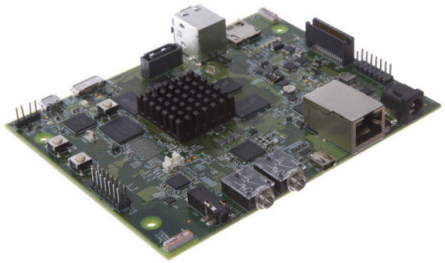
Xen

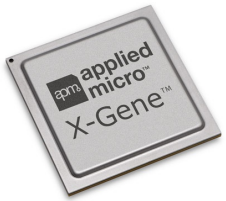
- Type-I hypervisor
- Support for ARM v7 and ARM v8 with virtualization extension

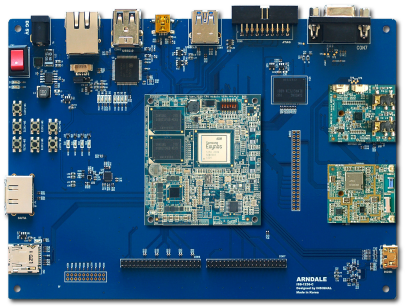
Example of hardware supported

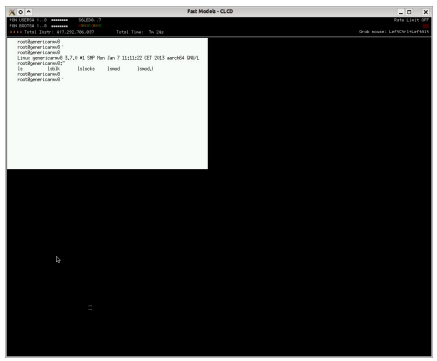




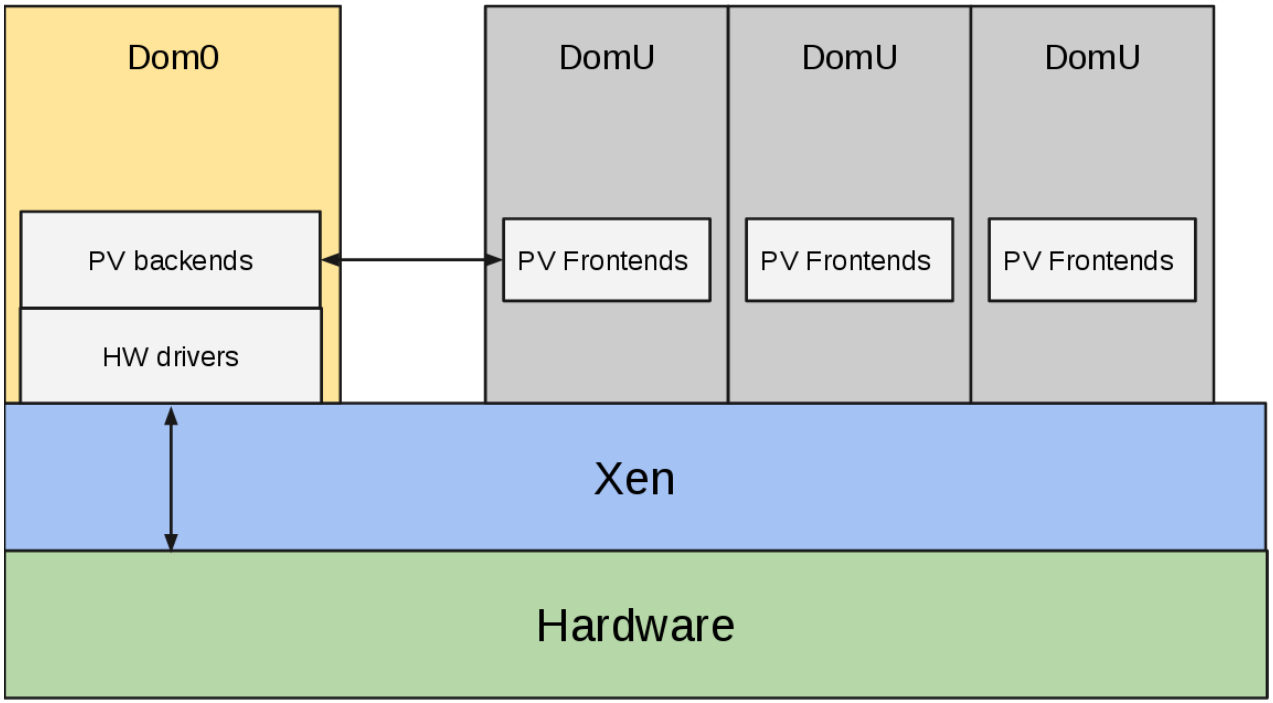




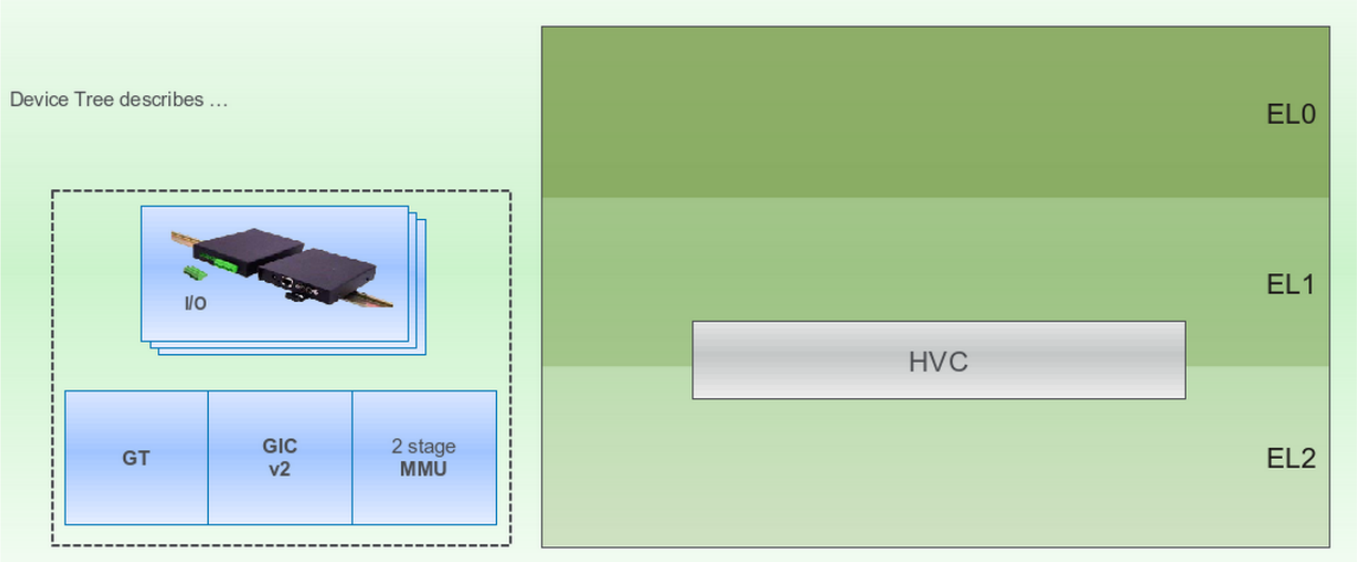




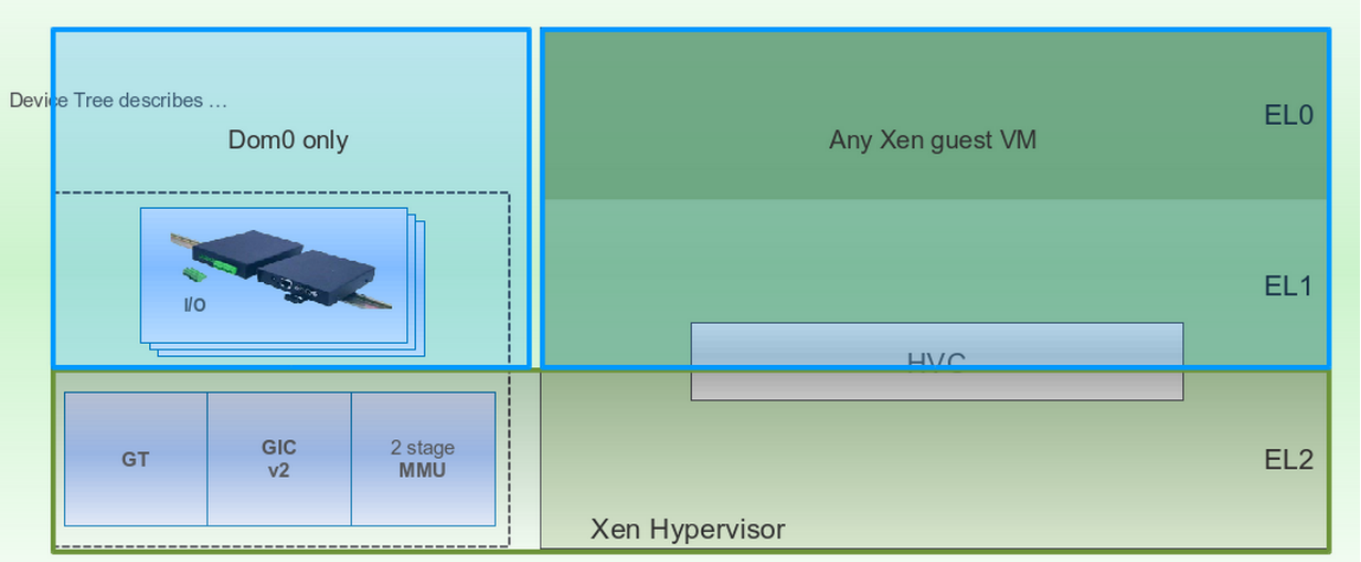
Xen architecture



ARM architecture



Xen on ARM architecture



DOM0

- First guest to start
- Nearly every devices are assigned to DOM0
  - Serial, IOMMU, Timer and GIC are used by Xen
  - Some devices can be blacklisted by Xen
- DOM0 kernel should use the Device Tree to discover the hardware

Requirements

- Guest boot ABI
- Device tree support
- Specific memory attribute
- Xen PV drivers
- Copy of *xen/include/public*

Guest boot ABI

Interface of the virtual machine:

- Loaders:
  - UEFI image
  - Linux zImage/Image
  - See *xc_dom_loader* structure in libxc to add a new loader
- Use of PSCI to bring up secondary CPUs

UEFI Image

- Only supported for AArch64 guest
- Require to build UEFI firmware for DOMU
  - tianocore edk II
  - *https://wiki.linaro.org/LEG/UEFIforXEN*
- No difference with baremetal after the firmware is booted
- Specification *http://www.uefi.org/specifications*

Linux zImage/Image

- Kernel loaded at a 4K aligned-address
- Specific values on some registers
  - ARM 32 bits
    - *r0 = 0*
    - *r1 = 0xffffffff*
    - *r2 = Device Tree physical address*
  - ARM 64 bits
    - *x0 = Device Tree physical address*
    - *x1 ... x4 = 0*
- MMU disabled
- Data cache disabled
- Instruction cache in an unknown state

Device Tree: What is it?

- Tree data structure with nodes that describes the physical devices in a system
- Working group to standardize device core bindings
  - *devicetree@vger.kernel.org*

Device Tree provided by Xen

- Basic Device Tree generated by the toolstack which contains:
  - CPUs
  - Memory
  - Timer
  - GIC
  - Hypervisor
  - PSCI
- Guest must support Device Tree in order to run on every Xen version

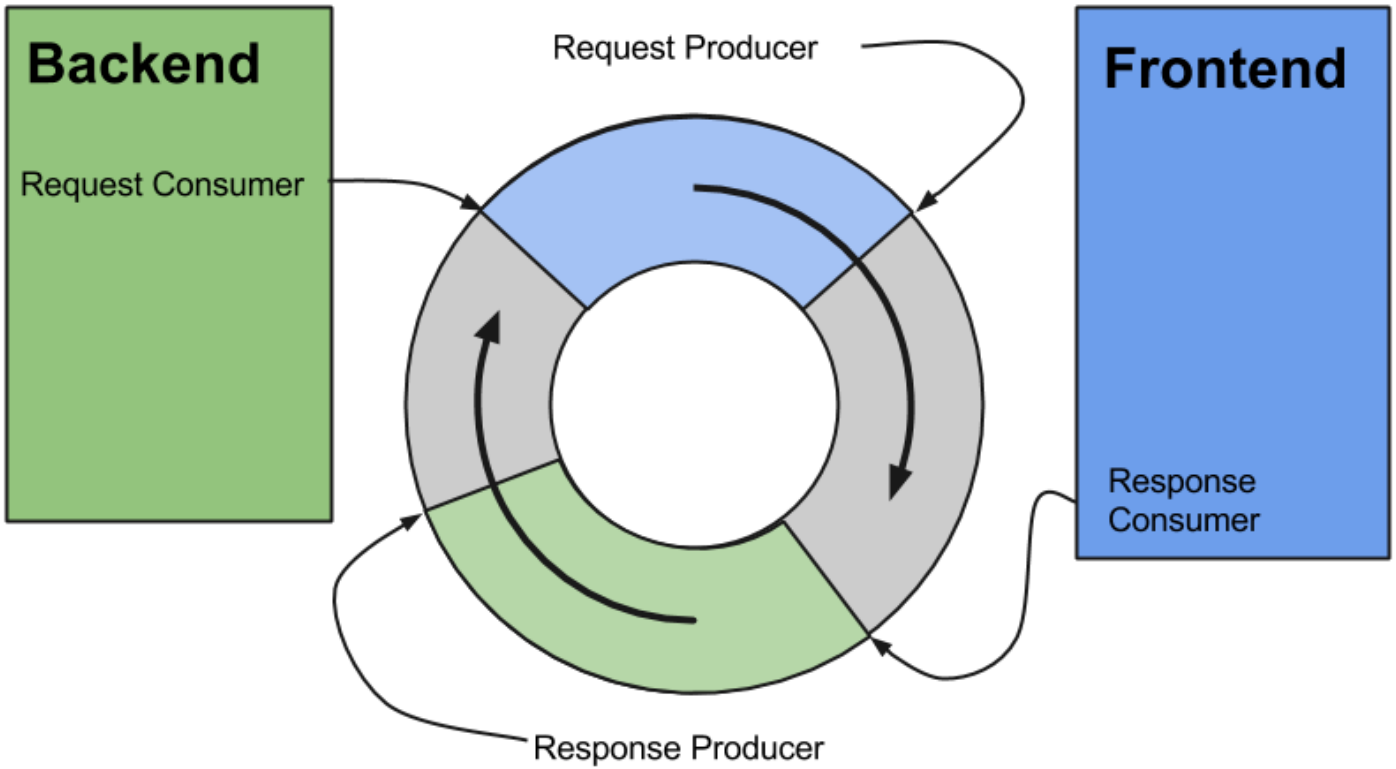
Memory

- Memory layout not fixed
  - Differ between Xen 4.4 and Xen 4.5
  - Documented in *xen/include/public/arch-arm.h*
- At least one RAM region will be located under 4GB

MMU and Memory

- MMU-less and data cache disabled guests are supported except for hypercall-based IO
- RAM restriction when MMU is enabled:
  - *3GB* when short-descriptor is used
  - *1TB* when LPAE is used
- RAM attribute should be *Write-Through* or *Write-Back*

PV protocol


Backend
Request Consumer
Request Producer
Frontend
Response Consumer
Response Producer

Hypercalls

- Use of **HVC** instruction
  - Hypercall tag *XEN_HYPERCALL_TAG = 0xEA1*
- *xen/public/arch-arm.h* provides hypercalls convention
  - ARM 32 bits
    - *r0 ...r4* used to store arguments
    - hypercall number is passed in *r12*
    - return value is in *r0*
  - ARM 64 bits
    - *x0 ... x4* used to store arguments
    - hypercall number is passed in *x16*
    - return value is in *x0*
- hypercall assembly wrappers provides in Linux
  - *arch/arm{,64}/xen/hypercall.S*
  - Dual-license GPLv2 and BSD
- memory hypercall based on 4K page granularity

Xen PV drivers

- Xen core architecture
  - Xenstore
  - Grant-Table
  - Event-channel
- Xen device drivers
  - Console
  - Block
  - Network
  - Framebuffer
- Drivers already available under:
  - BSD license in FreeBSD/Mini-OS
  - Dual-license GPLv2 and BSD in Linux

Debugging the guest kernel

- **xenctx** *domid vcpuid*
  - Get the state of a VCPU
  - To use in DOM0
- Xen console
  - Switch from DOM0 console to Xen console via *CTLR-a* three times
  - Useful key
    - *q* Domains information
    - *e* Event channel informations
    - *R* Reboot the machine (useful for DOM0 debugging)

Using Xen debugging facilities in the kernel

Use of **HVC** *0xFFxx*

- Supported when Xen is compiled with *debug=y*
- Requires to modify the guest

0xFFEX	Print the register *rX/xX*
0xFFFD	Print the program counter
0xFFFE	Print the character stored in *r0/x0*
0xFFFF	Dump the state of the VCPU

FreeBSD on Xen

- Support for x86 PVHVM and PVH
- Support for PVH DOM0 on x86
- Support of ARM 32 and ARM 64 architecture
- Device Tree bindings compatible with the Linux ones
- On going work to support Xen for ARM
  - Less than **500** lines of Xen arch specific code
  - PV drivers shared with x86

Xen PV drivers

- Update interface headers to Xen 4.4
  - FreeBSD is based on Xen 4.2 headers
  - ARM interface was not set in stone
- Drivers common with x86
  - Use the right xen type (xen_pfn_t, xen_ulong_t,...)
  - Support for HVM in console drivers
- Rework event channel handling
  - was x86 specific
  - still missing features
    - suspend/resume
    - pirq
  - common interrupt framework?

ARM 32

- New kernel config **XENHVM** created
- Only support for guest with 1 VCPU
- Loader:
  - Missing support using Device Tree with Linux boot ABI
  - zImage support?

ARM 64

- Xen enabled in the **GENERIC** kernel config
- Driver to parse Xen DT bindings added
- No modification necessary in the loader
  - UEFI based image
  - Kernel loaded from the Filesystem
  - Device Tree provided by UEFI
- Missing SMP bring up based on PSCI

OS supported by Xen

Out-of-box

MIRAGE OS
A Cloud Operating System

Future support

Questions?

- Xen devel ML: `xen-devel@lists.xenproject.org`
- Xen user ML: `xen-user@lists.xenproject.org`
- **#xenarm** on freenode

Fin