Xen on
ARM Cortex A15

Stefano Stabellini

Why?

NOKIA
NOKIA

smartphones: getting smarter

Quad-core
1.4 GHz
Cortex-A9

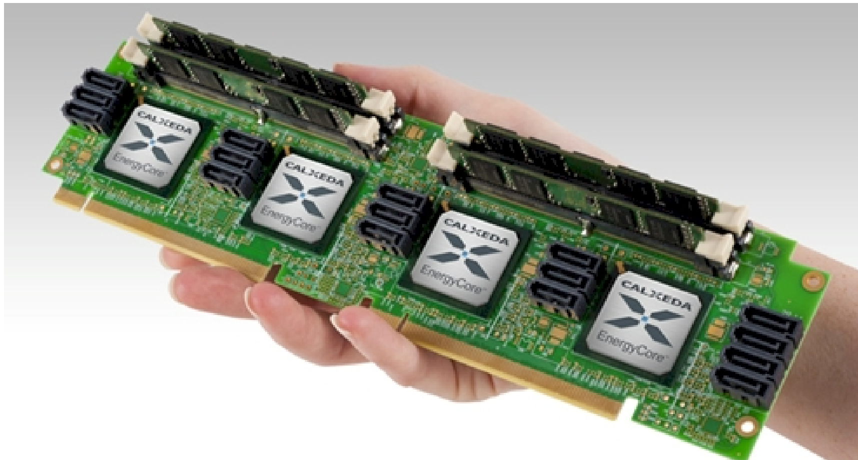
ARM Servers coming to market

4GB RAM, 4 cores per node

3 x 6 x 4 x 4 = 288 cores

single node virtualization -

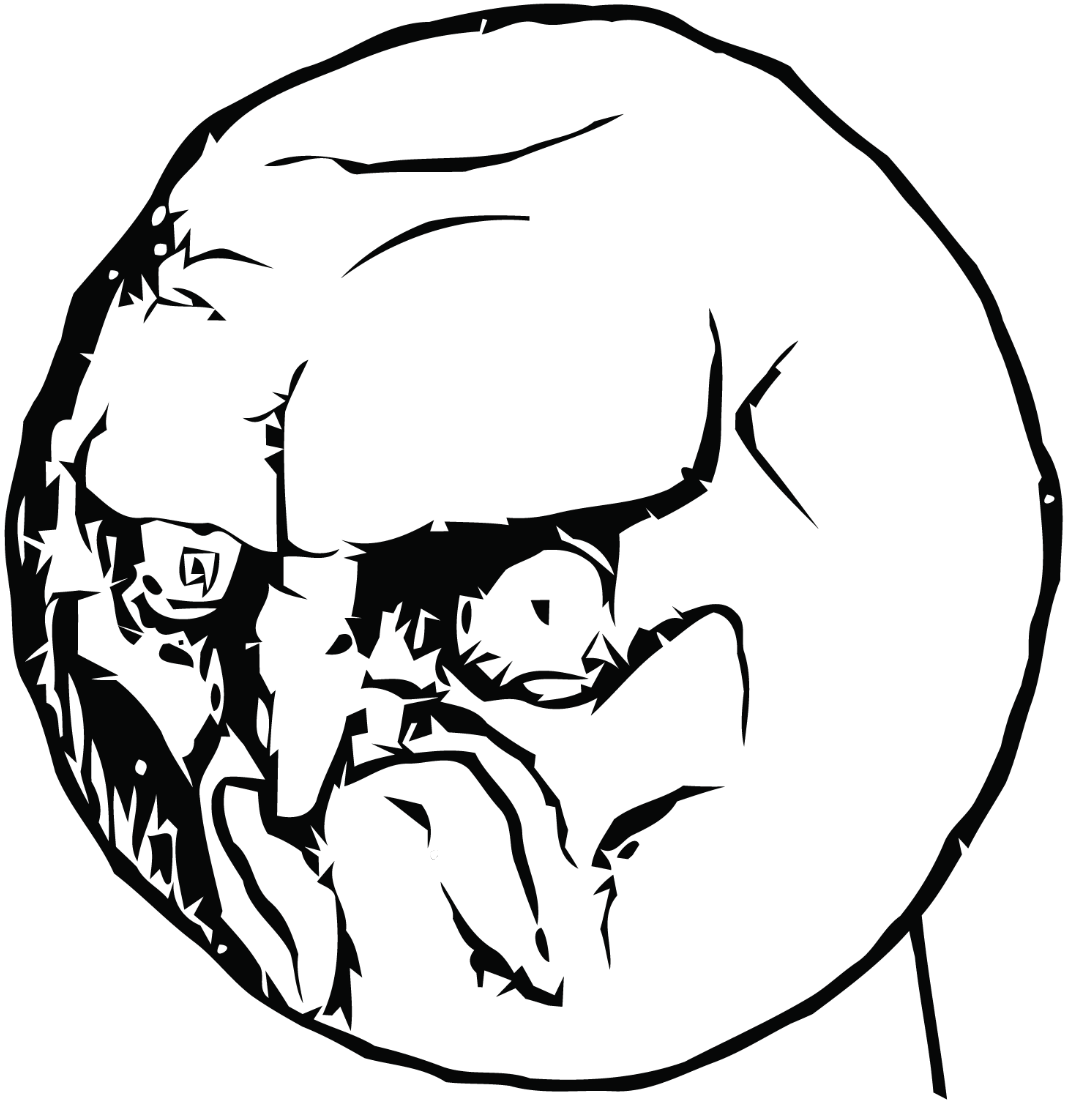
manageability -



Challenges

Another PVOPs infrastructure for ARM in Linux?

How would the Linux Community react?



NO.

ARMv7 virtualization extensions to the rescue!

"The ARM Architecture virtualization extension and Large Physical Address Extension (LPAE) enable the efficient implementation of virtual machine hypervisors for ARM architecture compliant processors."

Design goals

- exploit the hardware as much as possible
- one type of guest
  - no PVOPs
  - use PV interfaces for IO
- Rearchitected for the modern age:
  - no QEMU
  - no compat code
  - no shadow pagetables

One type of guest to rule them all

One type of guest

Like PV guests do it:

- support booting from a supplied kernel
- no emulated devices
- use PV interfaces for IO

↓

no need for QEMU

One type of guest

Like HVM guests do it:

- no PV MMU calls: exploit HW nested paging
- same entry point on native and on Xen
- use Device Tree to discover Xen presence
- no unnecessary devices in the Device Tree
- simple device emulation can be done in Xen

↓

no need for QEMU

Exploit the hardware

Exploit the hardware virtualization extensions support as much as possible:

- hypervisor mode
- MMU: second stage translation
    - no PV MMU calls: no need for PVOPs
    - no shadow pagetables: -10721 lines of code!!
- hypercall: HVC
- generic timer

General Interrupt Controller

an interrupt controller with virtualization support

- use the GIC to inject hardware interrupts into dom0
- use the GIC to inject event notifications into any guest domains with Xen support
    - use PPI 31
    - advertise the IRQ via Device Tree

- No special Xen entry point
- No Xen platform PCI device

The hypercall calling convention

the hypercall interface:

- **hvc** instruction
- hypervisor specific imm **0xEA1**
- hypercall arguments passed on registers

a 64 bit "ready" ABI

- a single hypercall ABI for 32 bit guests and 64 bit guests

↓

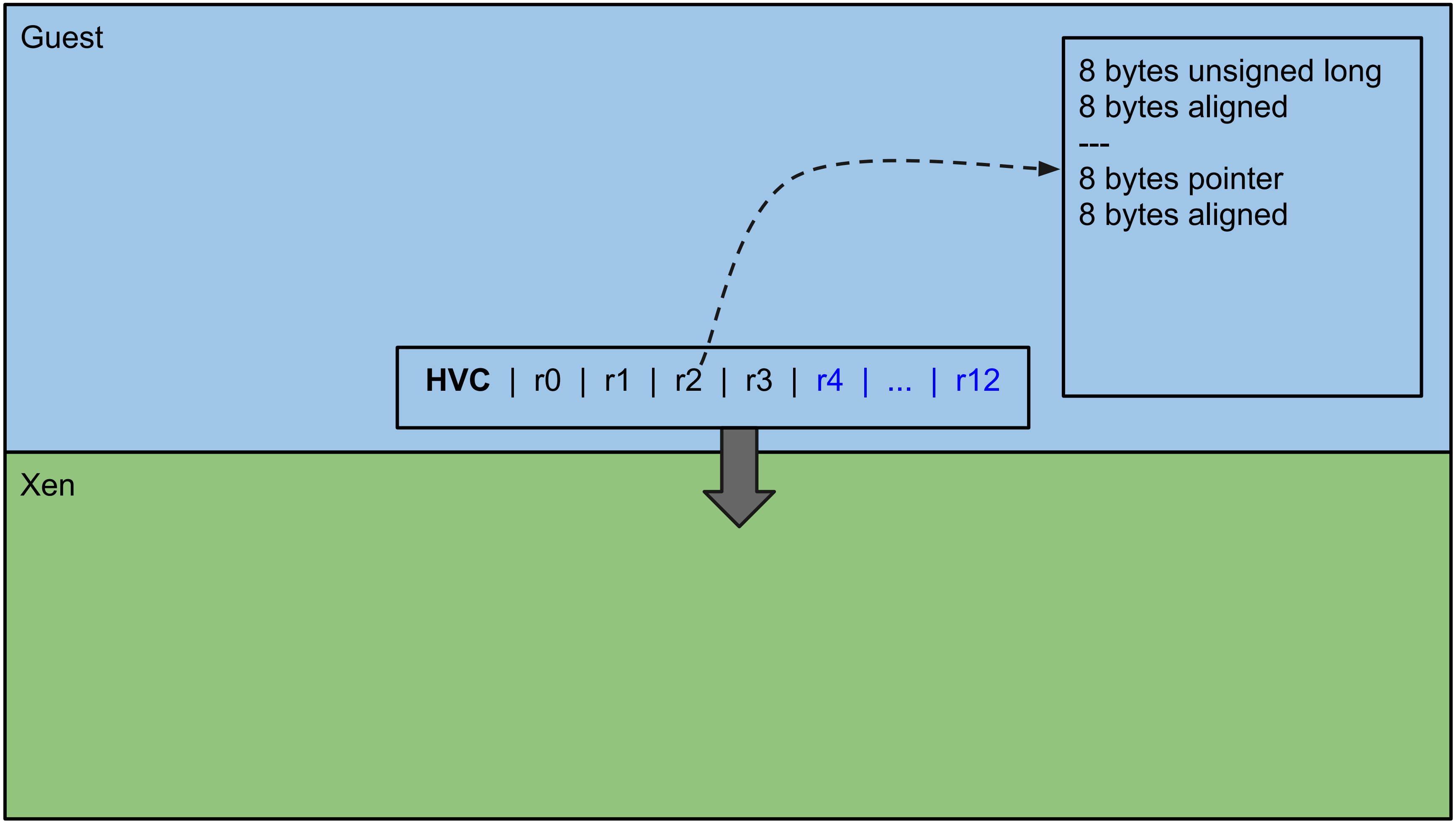
no compat code in Xen

  - ○ 2600 lines of code lighter

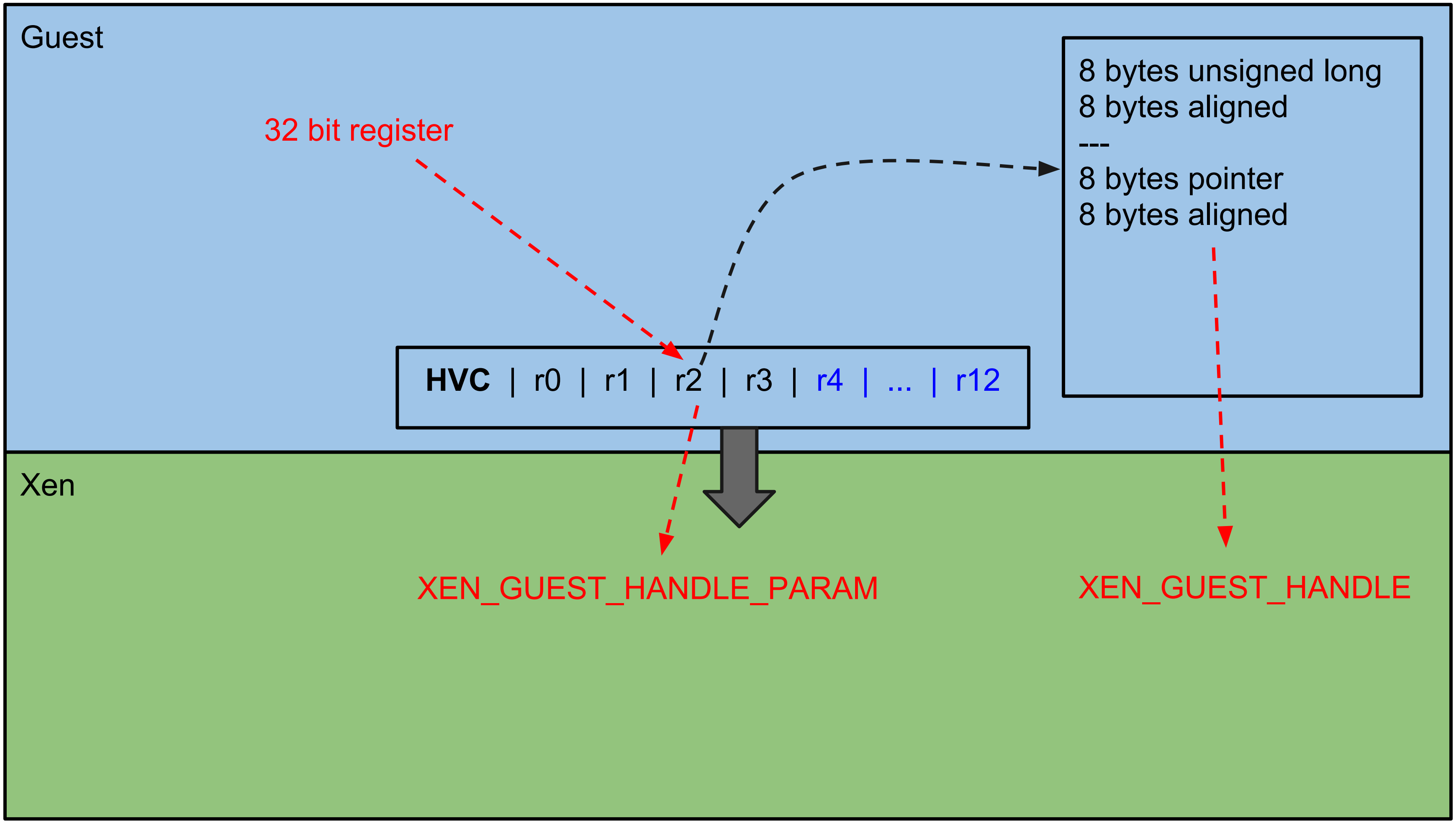
a 64 bit "ready" ABI

make unsigned long and pointers 8 bytes sized and 8 bytes aligned everywhere

a 64 bit "ready" ABI



a 64 bit "ready" ABI



a 64 bit "ready" ABI

make unsigned long and pointers 8 bytes sized and 8 bytes aligned everywhere

Collateral damage: a 1547 lines patch to

s/XEN_GUEST_HANDLE/XEN_GUEST_HANDLE_PARAM/

Device Tree

Use Device Tree to describe the virtual platform

```
hypervisor {
        compatible = "xen,xen", "xen,xen-4.2";
        reg = <0xb0000000 0x20000>;
        interrupts = <1 15 0xf08>;
};
```

Device Tree

Use Device Tree to describe the virtual platform

version of the Xen ABI

```
hypervisor {
        compatible = "xen,xen", "xen,xen-4.2";
        reg = <0xb0000000 0x20000>;
        interrupts = <1 15 0xf08>;
};
```

Grant table memory area

event notifications IRQ

Design goals:
did we meet them?

- exploit the hardware as much as possible

- one type of guest
  - no PVOPs
  - use PV interfaces for IO

- Rearchitected for the modern age:
  - no QEMU
  - no compat code
  - no shadow pagetables

Design goals: did we meet them?

- exploit the hardware as much as possible

- one type of guest
  - no PVOPs ← nested paging in HW, same entry point as nativ
  - use PV interfaces for IO ← no device emulation, use DT to describe the HW

- Rearchitected for the modern age:
  - no QEMU ← no device emulation, use DT to describe the HV
  - no compat code ← 64 bit ready ABI
  - no shadow pagetables ← nested paging in HW

Status of the Project

- Xen and Dom0 booting
- VM creation and destruction
- PV console, disk and network working
- Xen ARM patches mostly upstream
- Linux Xen ARM patches v3 sent to LKML

Open Questions: ACPI

"Modern PCs are horrible. ACPI is a complete design disaster in every way. But we're kind of stuck with it."

Linus Torvalds

- ACPI? Really??
- What about Device Tree?
- Do we need an ACPI parser in Xen?

  drivers/acpi: 110418 lines of code!
  Equivalent to 38% of the Xen (x86 and ARM) code base!!

Open Questions: UEFI

"EFI is this other [...] brain-damage (the first one being ACPI). "
Linus Torvalds

- Xen as Portable Executable
- Grub2 on ARM: multiboot2?
- UEFI runtime services
    - PVOPS? Trap and emulate?

Open Questions: Client Devices

- lack of a reference architecture
- UEFI Secure Boot
- Windows 8

Fim.