Xen on ARM

Stefano Stabellini
Virtualization: why it matters

Percentage of x86-Architecture Workloads Running in VMs

Source: Gartner (March 2011)
Xen: the gears of the cloud

- large user base
  more than 10 million individuals users

- power the largest clouds in production

- not just for servers
Xen: Open Source

GPLv2 with DCO (like Linux)
Diverse contributor community
source: Mike Day
http://code.ncultra.org
Xen Architecture

- Dom0
  - PV backends
  - HW drivers

- DomU
  - PV Frontends

- DomU
  - PV Frontends

- DomU
  - PV Frontends

Xen

Hardware
Xen Architecture: driver domains
Xen: advantages

- small surface of attack
- isolation
- resilience
- specialized algorithms (scheduler)
Xen Architecture: HVM guests

- Dom0
  - QEMU
  - PV backends
  - HW drivers

- HVM DomU

- stubdom
  - PV Frontends

- HVM DomU

Xen

Hardware
Xen upstream status

- Xen (Dom0 and DomU support, PV frontends and backends) fully upstream in Linux since v3.0
  A single 3.0.0 Linux kernel image boots on native, on Xen as domU, as dom0 and PV on HVM guest

- Xen upstream in QEMU since v1.3

- Xen supported by SuSE, Debian, Ubuntu, Fedora, CentOS, NetBSD and more
ARM Servers coming to market

4GB RAM, 4 cores per node
3 x 6 x 4 x 4 = 288 cores

single node virtualization -
manageability -
Design goals

● exploit the hardware as much as possible

● one type of guest

● Rearchitected for the modern age:
  ○ no QEMU
  ○ no compat code
  ○ no shadow pagetables
  ○ no PV MMU hypercalls
Xen on ARM architecture
Xen on ARM architecture

Device Tree describes …

Dom0 only

Any Xen guest VM

GT
GIC v2
2 stage MMU

Xen Hypervisor

EL0
EL1
EL2

HVC
Exploit the hardware extension virtualization

Exploit the hardware virtualization extensions support as much as possible:

- hypervisor mode
- MMU: second stage translation
  - no PV MMU calls
  - no shadow pagetables: -10721 lines of code!!
- hypercall: HVC
- generic timers
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
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:
- 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
The hypercall calling convention

the hypercall interface:

- **hvc** instruction
- hypervisor specific imm **0xEA1**
- hypercall arguments passed in registers
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>;
};
Use Device Tree to describe the virtual platform

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

- version of the Xen ABI
- Grant table
- memory area
- event notifications IRQ
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
ARMv8

● Builds on foundations laid by ARMv7
  ○ xen/arch/arm mostly common code

● Initially 32 bit dom0+domU on 64
  ○ Kernels already ready
  ○ 64-bit guest support in progress
## Code size

sometimes smaller is better

<table>
<thead>
<tr>
<th></th>
<th>Common</th>
<th>ARMv7</th>
<th>ARMv8</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>xen/arch/arm</td>
<td>5,122</td>
<td>1,969</td>
<td>821</td>
<td>7,912</td>
</tr>
<tr>
<td>C</td>
<td>5,023</td>
<td>406</td>
<td>344</td>
<td>5,773</td>
</tr>
<tr>
<td>ASM</td>
<td>99</td>
<td>1,563</td>
<td>477</td>
<td>2,139</td>
</tr>
<tr>
<td>xen/include/asm-arm</td>
<td>2,315</td>
<td>563</td>
<td>666</td>
<td>3,544</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>7,437</td>
<td>2,532</td>
<td>1,487</td>
<td>11,456</td>
</tr>
</tbody>
</table>

- Entire hypervisor ~200,000LOC
  - X86 (64-bit only) ~100,000LOC (~4,000 ASM)
    - ~22,000: HVM. ~14,000 MMU
Challenges

From the emulator to real hardware:
From the emulator to real hardware:

- barriers and flushes
- cache coherency
- GIC and race conditions
- virt_timer documentation bugs
Porting Xen to a new board

- Xen only relies on GIC and GT
- platform specific code in Xen is reduced to:
  - secondary cpus bring up
  - UART drivers
  - any platform specific bootup quirks (ideally none)
Status of the Project: ARMv7

- Xen and Dom0 booting on Versatile Express Cortex A15 and Arndale
- XL (Xen toolstack) ported to ARM
- PV console, disk and network working
- basic VM lifecycle operations functional
- Xen and Linux ARM patches fully upstream
Status of the Project: ARMv8

- Xen booting 64 bit
- Dom0 32 bit boots on Xen 64 bit
- 32 bit guest creation and destruction
- Shared code means most features developed on ARMv7 Just Work
Roadmap

Xen 4.3
○ ARMv7 (VExpress and Arndale) fully supported
○ ARMv8 64-bit port of the hypervisor

Xen 4.4
○ increase HCL
○ automated testing
○ ARMv8 64-bit virtual machines and tools
○ PCI passthrough, live migration

Linux 3.11/3.12
○ full ARMv8 64-bit Xen guest support
Demo
More Information

- http://www.xen.org
- Xen on ARM @wiki.xen.org: goo.gl/FKNXe