past, present and future

Stefano Stabellini
Xen architecture: PV domains
Xen arch: driver domains
- small surface of attack
- isolation
- resilience
- specialized algorithms (scheduler)
Xen and the Linux kernel

Xen was initially a university research project

- invasive changes to the kernel to run Linux as a PV guest
- even more changes to run Linux as dom0
Xen and the Linux kernel

Xen support in the Linux kernel not upstream

Great maintenance effort on distributions

Risk of distributions dropping Xen support
Xen and the Linux kernel

- PV support went in Linux 2.6.26
- basic Dom0 support went in Linux 2.6.37
- Netback went in Linux 2.6.39
- Blkback went in Linux 3.0.0

A single 3.0.0 Linux kernel image boots on native, on Xen as domU, as dom0 and PV on HVM guest
Xen and Linux distributions

2010
- Fedora and Ubuntu dropped Xen support from their Linux kernels
- Debian, Suse, Gentoo still provide Xen kernels
- XenServer went Open Source with XCP

Present
- Fedora and Ubuntu are adding Xen support back in kernel in the next releases
Xen architecture: HVM domains
Xen architecture: stubdoms
Xen and Qemu

- initially forked in 2005
- updated once every few releases
- Xen support went in upstream Qemu at the beginning of 2011
- Upstream Qemu is going to be used as device model with Xen 4.2
New developments: Libxenlight

Multiple toolstacks:
  - Xend, Xapi, XenVM, LibVirt, ...
  - code duplications, inefficiencies, bugs, wasted efforts

Xend:
  - difficult to understand, modify and extend
  - significant memory footprint
Libxenlight

What is Libxenlight:
- a small lower level library in C
- simple to understand
- easy to modify and extend

Goals:
- provide a simple and robust API for toolstacks
- create a common codebase to do Xen operations
- the unit testing tool for libxenlight
- feature complete
- a minimal toolstack
- compatible with xm

Do more with less!
XL: design principles

- smallest possible toolstack on top of libxenlight
- stateless

CLI → XL → libxenlight → EXIT
XL vs. Xend

XL: pros
- very small and easy to read
- well tested
- compatible with xm

Xend: pros
- provide XML RPC interface
- provide ”managed domains”
Libxenlight: the new world
paravirtualized interfaces in HVM guests
Linux as a guests: problems

Linux PV guests have limitations:
- difficult “different” to install
- limited set of virtual hardware

Linux HVM guests:
- install the same way as native
- very slow
Linux PV on HVM: the solution

- install the same way as native
- PC-like hardware
- access to fast paravirtualized devices
- exploit nested paging
Initial version in Linux 2.6.36:

- introduce the xen platform device driver
- add support for HVM hypercalls, xenbus and grant table
- enables **blkfront**, **netfront** and **PV timers**
- add support to PV suspend/resume
- the **vector callback** mechanism
Old style event injection

- DomU
  - BlkFront
  - NetFront
  - Xen Platform PCI Driver
    - Event
  - PCI Interrupt

- Dom0
  - BlkBack
  - NetBack
    - EVTCHNOP_send
  - EVTCHNOP_send
do_IRQ

  handle_fasteoi_irq

  handle_irq_event

  xen_evtchn_do_upcall

  ack_apic_level ← >=3 VMEXIT
The new vector callback
Receiving a vector callback

xen_evtchn_do_upcall
Linux PV on HVM: newer feats

Later enhancements (2.6.37+):
- ballooning
- PV spinlocks
- PV IPIs
- Interrupt remapping onto event channels
- MSI remapping onto event channels
Interrupt remapping
MSI remapping
<table>
<thead>
<tr>
<th></th>
<th>HVM guests</th>
<th>Classic PV on HVM</th>
<th>Enhanced PV on HVM</th>
<th>Hybrid PV on HVM</th>
<th>PV guests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot sequence</td>
<td>emulated</td>
<td>emulated</td>
<td>emulated</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>Memory</td>
<td>hardware</td>
<td>hardware</td>
<td>hardware</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>Interrupts</td>
<td>emulated</td>
<td>emulated</td>
<td>paravirtualized</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>I/O</td>
<td>emulated</td>
<td>emulated</td>
<td>paravirtualized</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>Disk</td>
<td>emulated</td>
<td>paravirtualized</td>
<td>paravirtualized</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>Network</td>
<td>emulated</td>
<td>paravirtualized</td>
<td>paravirtualized</td>
<td></td>
<td>paravirtualized</td>
</tr>
<tr>
<td>Privileged</td>
<td>hardware</td>
<td>hardware</td>
<td>hardware</td>
<td></td>
<td>paravirtualized</td>
</tr>
</tbody>
</table>
Benchmarks: the setup

Hardware setup:

Dell PowerEdge R710
CPU: dual Intel Xeon E5520 quad core CPUs @ 2.27GHz
RAM: 22GB

Software setup:

Xen 4.1, 64 bit
Dom0 Linux 2.6.32, 64 bit
DomU Linux 3.0 rc4, 8GB of memory, 8 vcpus
PCI passthrough of an Intel Gigabit NIC
CPU usage: the lower the better:
Kernbench

Results: percentage of native, the lower the better
Results: percentage of native, the lower the better
Results: percentage of native, the higher the better
Iperf tcp

Results: gbit/sec, the higher the better
Conclusions

PV on HVM guests are very close to PV guests in benchmarks that favor PV MMUs.

PV on HVM guests are far ahead of PV guests in benchmarks that favor nested paging.