Benefits of the new Xen paravirtualization mode

Roger Pau Monné

Brussels – February 2-3, 2013
Xen Architecture

- Xen Hypervisor
- Hardware device model (qemu)
- Control Domain: NetBSD or Linux
- toolstack
- Hardware Drivers
- netback blkback
- Paravirtualized (PV) Domain: NetBSD or Linux
- netfront blkfront
- Fully Virtualized (HVM) Domain: Windows, FreeBSD...

Xen Hypervisor

Hardware
- I/O Devices
- CPU
- Memory
Paravirtualization

- Virtualization technique developed in the late 90s
- Designed by:
  - XenoServer research project at Cambridge University
  - Intel
  - Microsoft labs
- x86 instructions behave differently in kernel or user mode, options for virtualization were full software emulation or binary translation.
  - Design a new interface for virtualization
  - Allow guests to collaborate in virtualization
  - Provide new interfaces for virtualized guests that allow to reduce the overhead of virtualization
- The result of this work is what we know today as paravirtualization
Paravirtualization

- All this changes lead to the following interfaces being paravirtualized:
  - Disk and network interfaces
  - Interrupts and timers
  - Boot directly in the mode the kernel wishes to run (32 or 64 bits)
  - Page tables
  - Privileged instructions
The paravirtualization model works well on 32bit systems, but it has problems on 64bits

- AMD decided to remove segmentation limit in 64bits, that was used by Xen to protect hypervisor memory
- The 64bit architecture only has two memory protection levels
- This is bad for the Xen architecture, three protection levels are needed in order to provide isolation between the hypervisor, the guest kernel and the guest userspace.
- In 64bit PV guests every system call bounces up to Xen that performs the context switch to the guest kernel.
Paravirtualization

- Pros of paravirtualization
  - Provides near to bare metal speed
  - Reduces overhead of virtualization
- Cons of paravirtualization
  - Requires heavy modifications to the guest OS
With the introduction of hardware virtualization extensions, Xen is able to run unmodified guests. This requires emulated devices, which are handled by Qemu. Makes use of nested page tables when available.
Full virtualization

- Pros of full virtualization
  - Doesn’t require guest OS changes

- Cons of full virtualization
  - Slow IO because of emulated devices
  - Legacy boot
  - Need to run Qemu for each guest
The full virtualization spectrum

<table>
<thead>
<tr>
<th>VS</th>
<th>Software virtualization</th>
<th>Poor performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>VH</td>
<td>Hardware virtualization</td>
<td>Room for improvement</td>
</tr>
<tr>
<td>PV</td>
<td>Paravirtualized</td>
<td>Optimal performance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk and network</th>
<th>Interrupts and timers</th>
<th>Emulated motherboard and page tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>HVM</td>
<td>VS</td>
<td>VS</td>
</tr>
<tr>
<td>HVM with PV drivers</td>
<td>PV</td>
<td>VS</td>
</tr>
<tr>
<td>PVHVM</td>
<td>PV</td>
<td>PV</td>
</tr>
<tr>
<td>PV</td>
<td>PV</td>
<td>PV</td>
</tr>
</tbody>
</table>

Brussels – February 2-3, 2013

Benefits of the new Xen paravirtualization mode
Guest support in BSD

- List of BSD systems and operation modes:

<table>
<thead>
<tr>
<th></th>
<th>PV</th>
<th>PVHVM</th>
<th>HVM with PV drivers</th>
<th>HVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetBSD</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>FreeBSD</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>OpenBSD</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>DragonflyBSD</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

- Main problems:
  - Only PV guests are allowed as control domains (Dom0)
  - Some providers will only allow PV guests, or charge extra for HVM guests
Evolving Xen PV

- When PV was designed there were no virtualization extensions in hardware
- Everything is done using PV interfaces
- Hardware virtualization can provide speed improvements and ease of implementation
- Previous slides show how HVM mode evolved to use PV components
- But what happens if PV evolves to use HVM components?
Evolving Xen PV

- **VS**: Software virtualization
- **VH**: Hardware virtualization
- **PV**: Paravirtualized

### Benefits of the New Xen Paravirtualization Mode

<table>
<thead>
<tr>
<th>Mode</th>
<th>Disk and network</th>
<th>Interrupts and timers</th>
<th>Emulated motherboard</th>
<th>Privileged instructions and page tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>HVM</td>
<td>VS</td>
<td>VS</td>
<td>VS</td>
<td>VH</td>
</tr>
<tr>
<td>HVM with PV drivers</td>
<td>PV</td>
<td>VS</td>
<td>VS</td>
<td>VH</td>
</tr>
<tr>
<td>PVHVM</td>
<td>PV</td>
<td>PV</td>
<td>VS</td>
<td>VH</td>
</tr>
<tr>
<td>PVH</td>
<td>PV</td>
<td>PV</td>
<td>PV</td>
<td>VH</td>
</tr>
<tr>
<td>PV</td>
<td>PV</td>
<td>PV</td>
<td>PV</td>
<td>PV</td>
</tr>
</tbody>
</table>
PVH

- Major features:
  - Runs in Ring 0
  - Uses PV entry point, skips BIOS emulation
  - Uses PV event channels, no APIC emulation
  - Native Page tables
  - Native IDT

- Development done by Mukesh Rathor at Oracle

- Main focus is increasing performance in 64bit PV guests
Benefits for the BSD world:

- Uses PVHVM callbacks for events
- Doesn’t use the PV MMU
- Can be used as control domain (Dom0)

- The main benefit for BSD systems is that PVH is going to simplify one of the most difficult aspects of PV, the MMU

- Since the MMU is virtualized by hardware, porting new OSes to run in the PVH mode is greatly simplified
Conclusions

- Xen offers different guest virtualization modes
- Until now, PV was the only allowed mode for control domains
- PVH can be used as control domain, and simplifies the porting efforts
- The MMU changes, that where OS dependant are no longer necessary
- Drivers code can be shared between different BSD systems
- Transition from PVHVM to PVH is simpler than PV
Q&A

Thanks
Questions?