Agenda

- Client virtualization challenges
- Use cases
- Type 1 vs. Type 2 in clients
- I/O virtualization in clients
- Dynamic device assignment
- Future trends
Client hypervisor challenges

- User centered interactive experience
- Varied range of I/O devices
- Graphics, sound and internet are paramount to the experience
- Full ACPI support is required
- Endpoint is “in the wild”, VM isolation is a concern

Bottom line: endpoints weren’t designed to be virtualized
Virtualization technologies matrix

Type-1

Type-2

Emulated

Pass Through
Use cases

• Personal and Corporate environments side-by-side
• Endpoint lifecycle management
• Endpoint consolidation
• “Bring your own PC”
• Trusted computing
Objectives

• VM isolation
• Security
• Performance
• HW resource control
• Robustness
• Usability
Type-2 hypervisor

- Hardware
  - "Legacy" operating system
    - Drivers
    - Scheduler
    - Memory mgmt
  - "Legacy" operand system
    - Hypervisor (User process)
      - VM
      - VM
      - Emulated devices
  - User process
  - User process
  - User process

Hardware

5/19/09
Type-1 hypervisor (Xen based)

- Virtual Machine DomU
  - Using device pass-through
  - Drivers of physical devices
- Virtual Machine DomU
  - Drivers of emulated devices
- Domain-0
  - Drivers
- QEMU-dm
- Hypervisor
  - Scheduler
  - Memory mgmt
- Hardware

5/19/09
# Type-1 vs. Type-2

<table>
<thead>
<tr>
<th>Type-1 Hypervisor</th>
<th>Type-2 Hypervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs directly on the HW – “bare metal”</td>
<td>Runs as a user space process under a standard OS</td>
</tr>
<tr>
<td>Intrusive by nature</td>
<td>Non-intrusive</td>
</tr>
<tr>
<td>Deployment is a challenge</td>
<td>Simple to deploy</td>
</tr>
<tr>
<td>Total VM isolation</td>
<td>VMs (and hypervisor) susceptible to viruses/maleware infecting the OS</td>
</tr>
<tr>
<td>Owns the CPU, schedules VMs directly</td>
<td>Depends on the host OS scheduler</td>
</tr>
</tbody>
</table>
## Emulated vs. Pass-through I/O

<table>
<thead>
<tr>
<th>Emulated I/O</th>
<th>Pass-Through I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW independence, easy deployment and migration</td>
<td>Native drivers are required for the specific HW</td>
</tr>
<tr>
<td>Low performance regardless of HW</td>
<td>Full native performance</td>
</tr>
<tr>
<td>Restricted feature set, lowest common denominator</td>
<td>Full HW feature set, “get what you paid for”</td>
</tr>
<tr>
<td>Allows complex HW control scenarios with PV drivers (e.g. dynamic device assignment)</td>
<td>Restricted to OS/native driver behavior</td>
</tr>
</tbody>
</table>
Main challenges: DMA and interrupts

Hardware assisted:
- Intel VT-d (e.g. ICH9/PM45 chipset)
- AMD IOMMU

100s of millions of endpoints without I/O-V assisted HW are deployed.

Neocleus client hypervisor solution
- 1:1 memory mapping
- Interrupt sharing
Load the “primary” VM into its’ native memory region.
Neocleus interrupt sharing

Non-virtualized endpoint

OS
Driver
APIC

Devices with shared IRQ

Check device IC

Virtualized endpoint with passthrough

VM
Driver
APIC

Which VM should get IRQ?

Devices with shared IRQ
Dynamic device assignment

- Required in client virtualized environment
- Implementation for specific I/O-V techniques:

<table>
<thead>
<tr>
<th>I/O virtualization technique</th>
<th>Dynamic assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emulated I/O</td>
<td>PV driver + backend multiplexing</td>
</tr>
<tr>
<td>I/O over network</td>
<td>Network switch in Dom0</td>
</tr>
<tr>
<td>Passthrough with IOMMU</td>
<td>Switching I/O page tables</td>
</tr>
<tr>
<td>Passthrough without IOMMU (1:1 MM)</td>
<td>A real challenge...</td>
</tr>
</tbody>
</table>
ACPI Passthrough

• A must for a proper laptop experience:
  – Battery
  – Lid
  – AC
  – Sleep mode
  – Special function keys

• Solution:
  – decompile vendor DSDT and integrate into the VM DSDT
  – Use ACPI in Dom0 to trap ACPI events and pass them to the VM
CPU virtualization

• Standard OS scheduler
  – has knowledge of process “profile”
  – Can penalize CPU bound processes and credit I/O bound processes
  – The result is a more responsive user interaction

• Hypervisor
  – Sees an entire VM as a single process
  – Can’t discern I/O bound and CPU bound processes
  – Can’t even discern an idle OS process
  – Need to schedule the I/O back end in Dom0 as well
VM isolation

• Why?
  – Robustness requires that no one VM may crash the endpoint
  – Security mandates that compromised VMs can’t contaminate other VMs

• How?
  – VMs memory mgmt utilize shadow page tables
  – PT implementations must support dynamic assignment of critical devices
  – I/O passthrough with 1:1 mapping susceptible to DMA attacks - still a challenge...
Future trends

• SR-IOV
• Widespread deployment of IOMMU
• Nested VMs
• Windows switchable graphics
• Commoditized hypervisor
Questions?

Contact us: info@neocleus.com