Software Techniques for Avoiding Hardware Virtualization Exits

Ole Agesen    Jim Mattson    Radu Rugina    Jeffrey Sheldon

VMware
Server Virtualization

% virtualized workloads worldwide

Source: IDC Worldwide Virtualization Tracker
**x86 Virtualization**

- **Virtual Machine Monitor (VMM)** abstracts physical hardware

- Two approaches:
  - Binary Translation (BT)
  - Hardware-Assisted (HV) (Intel VT-x and AMD-V)
Hardware-Assisted x86 Virtualization

Guest execution using
Hardware Virtualization (HV)

Virtual Machine Monitor (VMM)

exit
resume
Virtualization Exits Are Expensive

<table>
<thead>
<tr>
<th>Microarchitecture</th>
<th>Launch Date</th>
<th>Hardware Exit + Resume (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prescott</td>
<td>2005</td>
<td>3963</td>
</tr>
<tr>
<td>Merom</td>
<td>2006</td>
<td>1579</td>
</tr>
<tr>
<td>Penryn</td>
<td>2008</td>
<td>1266</td>
</tr>
<tr>
<td>Nehalem</td>
<td>2009</td>
<td>1009</td>
</tr>
<tr>
<td>Westmere</td>
<td>2010</td>
<td>761</td>
</tr>
<tr>
<td>Sandy Bridge</td>
<td>2011</td>
<td>784</td>
</tr>
</tbody>
</table>
This Talk: Cluster HV Exits

Guest execution using Hardware Virtualization (HV)

Virtual Machine Monitor (VMM)
This Talk: Cluster HV Exits

Guest execution using
Hardware Virtualization (HV)

Virtual Machine Monitor (VMM)
Outline

• Exit Pairs
• Exit Clusters
• Nested Virtual Machines
• Results
• Conclusions
Exit Pairs

- 32-bit Guest OS using Physical Address Extension (PAE)
- Shadow paging
- Page table entry updates use two 32-bit writes
- Each write causes an exit

```
Guest

...  

mov 4(%ecx), %esi
mov (%ecx), %ebx  

...  
```
Exit Pairs

Guest

... 

mov 4(%ecx), %esi

mov (%ecx), %ebx

...

VMM

VMM updates shadow page table entries
Exit Pairs

Upon exit:
- VMM inspects next instruction
- Detects access to adjacent byte
- Executes both instructions
Upon exit:
- VMM inspects next instruction
- Detects access to adjacent byte
- Executes both instructions
- Optimizes execution

Guest

...  

mov 4(%ecx), %esi

mov (%ecx), %ebx

...  

VMM
Exit Clusters

• Upon exit:
  • Scan a few (up to 16) instructions downstream
  • Identify exiting instructions
  • Form a cluster of instructions, executed all at once

• Challenges:
  • Cluster formation
  • Efficient execution
Cluster Formation

Guest

in %al, %dx

VMM

HV exit occurs
Cluster Formation

```
Guest

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>in %al, %dx</td>
</tr>
<tr>
<td>out 0x80, %al</td>
</tr>
<tr>
<td>mov %al, %cl</td>
</tr>
<tr>
<td>mov %dl, 0xc0</td>
</tr>
<tr>
<td>out %al, %dx</td>
</tr>
<tr>
<td>out 0x80, %al</td>
</tr>
<tr>
<td>xchg %ah, %al</td>
</tr>
<tr>
<td>xor %cl, %cl</td>
</tr>
</tbody>
</table>

VMM

HV exit occurs
Decode instructions
```
Cluster Formation

Guest

| in  %al, %dx |
| out 0x80, %al |
| mov %al, %cl |
| mov %dl, 0xc0 |
| out %al, %dx |
| out 0x80, %al |
| xchg %ah, %al |
| xor %cl, %cl |

VMM

- HV exit occurs
- Decode instructions
- Identify exiting instructions
Cluster Formation

Guest

in %al, %dx
out 0x80, %al
mov %al, %cl
mov %dl, 0xc0
out %al, %dx
out 0x80, %al

VMM

HV exit occurs
Decode instructions
Identify exiting instructions
Form cluster
Cluster Formation

Guest

| in %al, %dx |
| out 0x80, %al |
| mov %al, %cl |
| mov %dl, 0xc0 |
| out %al, %dx |
| out 0x80, %al |

VMM

HV exit occurs
Decode instructions
Identify exiting instructions
Form cluster

gap fillers
Cluster Formation

**Guest**

```
in %al, %dx
out 0x80, %al
mov %al, %cl
mov %dl, 0xc0
out %al, %dx
out 0x80, %al
```

**VMM**

HV exit occurs
- Decode instructions
- Identify exiting instructions
- Form cluster
- Execute and resume HV
Efficient Exit Handling

• **Cluster translation:**
  • Generate a translation for the cluster
  • Insert it in a Translation Cache (TC)
  • Specialize translations (e.g., on addressing mode)

• **Translation reuse:**
  • Compile once
  • Reuse for all subsequent exits
Cluster Translation

Guest

```
in %al, %dx
out 0x80, %al
mov %al, %cl
mov %dl, 0xc0
out %al, %dx
out 0x80, %al
```

Coherency Checks

Code Translation
## Dynamic Cluster Formation

### Guest

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>out %al, %dx</code></td>
</tr>
<tr>
<td><code>mov %cx, -0x12(%bp)</code></td>
</tr>
<tr>
<td><code>sub %si, %cx</code></td>
</tr>
<tr>
<td><code>mov %dl, 0x3c5</code></td>
</tr>
<tr>
<td><code>mov %al, 0x1</code></td>
</tr>
<tr>
<td><code>out %al, %dx</code></td>
</tr>
<tr>
<td><code>shr %bl</code></td>
</tr>
<tr>
<td><code>sbb %ah, %ah</code></td>
</tr>
<tr>
<td><code>shl %al</code></td>
</tr>
<tr>
<td><code>mov es:(%di), %ah</code></td>
</tr>
</tbody>
</table>
Dynamic Cluster Formation

Do memory accesses cause exits?

<table>
<thead>
<tr>
<th>Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>out %al, %dx</code></td>
</tr>
<tr>
<td><code>mov %cx, -0x12(%bp)</code></td>
</tr>
<tr>
<td><code>sub %si, %cx</code></td>
</tr>
<tr>
<td><code>mov %dl, 0x3c5</code></td>
</tr>
<tr>
<td><code>mov %al, 0x1</code></td>
</tr>
<tr>
<td><code>out %al, %dx</code></td>
</tr>
<tr>
<td><code>shr %bl</code></td>
</tr>
<tr>
<td><code>sbb %ah, %ah</code></td>
</tr>
<tr>
<td><code>shl %al</code></td>
</tr>
<tr>
<td><code>mov es:(%di), %ah</code></td>
</tr>
</tbody>
</table>
Dynamic Cluster Formation

Do memory accesses cause exits?

Sometimes.
Instruction Classification

• **Strongly exiting:**
  • Always cause exits
  • Examples: `in`, `out`, `cpuid`

• **Weakly exiting:**
  • Dynamic exiting behavior
  • Example: memory accesses
  • Runtime VMM support for detecting such cases
Dynamic Cluster Formation

<table>
<thead>
<tr>
<th>Guest</th>
<th>3</th>
<th>out %al, %dx</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>mov %cx, -0x12(%bp)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>sub %si, %cx</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>mov %dl, 0x3c5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>mov %al, 0x1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>out %al, %dx</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>shr %bl</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>sbb %ah, %ah</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>shl %al</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>mov es:(%di), %ah</td>
<td></td>
</tr>
</tbody>
</table>

Translation postponed
Count exits
Translate on 3rd exit
Dynamic Cluster Formation

Guest

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>out %al, %dx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>mov %cx, -0x12(%bp)</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>sub %si, %cx</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>mov %dl, 0x3c5</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>mov %al, 0x1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>out %al, %dx</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>shr %bl</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>sbb %ah, %ah</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>shl %al</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>mov es:(%di), %ah</td>
</tr>
</tbody>
</table>

Translation postponed
Count exits
Translate on 3rd exit
Cluster With Complex Control-Flow

0x02e5: IN %al,%dx
0x02e6: TEST %al,$0x8
0x02e8: JZ 0xb ;0x2f5

0x02ea: LOOP %cx,-0x7 ;0x2e5

0x02ec: DEC %bx
0x02ed: JNZ -0xa ;0x2e5

0x02ef: <CS> MOV 0x2e8,$0xeb

0x02f5: MOV %bx,$0x40
0x02f8: SUB %cx,%cx

0x02fa: IN %al,%dx
0x02fb: TEST %al,$0x8
0x02fd: JNZ 0xb ;0x30a

0x02ff: LOOP %cx,-0x7 ;0x2fa

0x0301: DEC %bx
0x0302: JNZ -0xa ;0x2fa
Nested Virtual Machines
Nested VMs

• Run hypervisor as a guest [Ben-Yehuda et al., OSDI’10]

• Simulate large-scale virtualized environments with fewer hosts

• Training, testing, debugging

• “Windows XP mode” in Win 7
HV for Nested VMs

- Outer VMM transitions
  - Handled in hardware
- Inner VMM transitions
  - Virtual hardware exits are emulated in software
HV for Nested VMs

- **Outer VMM transitions**
  - Handled in hardware

- **Inner VMM transitions**
  - Virtual hardware exits are emulated in software
  - Virtual software exit path has lots of exiting instructions (e.g., vmread, vmwrite)
HV for Nested VMs

• Outer VMM transitions
  • Handled in hardware

• Inner VMM transitions
  • Virtual hardware exits are emulated in software
  • Virtual software exit path has lots of exiting instructions (e.g., vmread, vmwrite)
  • 10x slowdown!
Exit Avoidance

• **Cluster inner exits**
  • Big impact due to high cost of inner exits

• **Cluster outer exits:**
  • Big impact due to the high frequency of outer exits
  • Opportunity: cluster-friendly VMM instruction scheduling
Experimental Results
Implementation

- Fully implemented in VMware products
- Workstation, Fusion, ESX
- Evolved over many years from exit pairs to complex clusters
- Validated by use in the field
Non-Nested VMs

- **PassMark** (2D graphics benchmark)
  - Clustering improves score by 50% - 80%
- **VMmark** (virtualization benchmark)
  - Consistent exit rate reduction
  - No measurable runtime improvement
- Netperf against a VM with virtual e1000 NIC
  - 24% reduction of exits per packet roundtrip
Nested VM Speedup

Kernel-compile workload in the inner VM

- Inner clusters only
- Outer clusters only
- Inner + outer clusters

AMD

Intel
Future Directions

• Support more complex clusters
  • Non-contiguous clusters
  • Exits in the middle of loops

• Optimize memory accesses in clusters
  • Cache and reuse work for accesses on the same page

• Applications to virtualized low-latency workloads
Conclusions

• HV exits are expensive, can sap performance
• Software exit clustering complements hardware optimizations for HV exits
• Clustering is key to enabling reasonable performance for nested VMs