Nested Virtualization

State of the art and future directions

Bandan Das
Yang Z Zhang
Jan Kiszka
Outline

- Introduction
- Changes and Missing Features for AMD
- Changes and Missing Features for Intel
- Working hypervisors and performance evaluation
- Discussion on other ongoing work
  - Migration Support
  - VT-d emulation
- Wrap-up and Questions
Introduction

- Nested Virtualization –
Introduction

• Uses

- Operating system hypervisors (Linux/KVM, WinXP mode in newer versions of Windows)
- Cloud Computing – Give users the ability to run their own hypervisors!
- Security – McAfee DeepSafe
- Testing/debugging hypervisors
- Interoperability
• How it works (on Intel)
  – \(L_0\) runs \(L_1\) with VMCS\(_{01}\)
  – \(L_1\) wants to run \(L_2\) and executes vmlaunch with VMCS\(_{12}\)
  – vmlaunch traps to \(L_0\)
  – \(L_0\) merges VMCS\(_{01}\) with VMCS\(_{12}\) to create VMCS\(_{02}\) and run \(L_2\)
  – If \(L_2\) traps, we are back in \(L_0\)
  – \(L_0\) decides whether to handle trap itself or forward to \(L_1\)
  – Eventually \(L_0\) resumes \(L_1\)
  – .....
Nested Virtualization - AMD

- Stable codebase
  - “nested” is enabled by default
- AMD-v
  - Advanced virtual Interrupt Controller (AVIC)
  - Hardware yet to arrive!
- More Testing
  - Hard to find bugs always exist!
  - Newer releases of common and new hypervisors
  - Nesting introduces I/O bottlenecks
- Are we spec compliant?
Nested Virtualization - Intel

• Recent Changes
  – Specification conformance
    • Additional error checks on emulated vmx functions
    • Corresponding tests in kvm-unit-tests
  – Intel Memory Protection Extensions
    • Bounds checking on memory references
    • VMX support: “clear BNDCFGS” and “BNDCFGS” VMCS exit controls and “BNDCFGS” VMCS field
    • Nested Support: Let $L_1$ hypervisor read and write the MPX controls($vmcs_{12}$->guest_bndcfgs)
  – Tracing improvements
Nested Virtualization - Intel

• Recent Changes
  - Interrupt Acknowledgement Emulation
  - Interrupt Injection Rework
    • Inspired by Jailhouse hypervisor
    • Also speeds up Windows execution (Complemented by TPR Shadow support)
Nested Virtualization - Intel

• Improve Stability
  – More testing
  – Nested vmx is still disabled by default!
  – The test matrix is quite complicated with so many configurations and hypervisors

• Are we specification compliant?
  – Also helps in identifying buggy hypervisors
Nested VPID

1. Virtual Processor Identifier
   - Tag address space and avoid a TLB flush
2. We don't advertise vpid to the L₁ hypervisor
3. L₀ uses the same vpid to run L₁ and all its guests
4. KVM flushes vpid when switching between L₁ and L₂
5. Advertise vpid and maintain a mapping for L₁'s vpids
Nested Virtualization - Intel

- MSR load/store
  - Hypervisor loads/saves a MSR list during VMENTER/VMEXIT
  - Mandatory according to specification

- Nested APIC-v
  - Reduce VMEXITs
  - Motivation: performance gains
• Test Environment
  - Host (L₀) – AMD Opteron(tm) Processor 6386 SE (16 cores), 32 GB RAM, Fedora 20
  - Qemu options to run L₁: `-cpu host -m 20G -smp 10`
  - Qemu options L₁ uses to run L₂: `-cpu qemu64 -m 8G -smp 8`

• Guest Status (L₁ hypervisor)
  - Linux (Fedora 20 64 bit)
  - Xen 4.4.3 running in Ubuntu 12.04
  - JailHouse
  - ESX
AMD Performance Evaluation

- Test Environment
  - Host: AMD Opteron(tm) Processor 6386 SE / 32 GB RAM
  - L₀, L₁ and L₂: Fedora 20
  - Kernel 3.17.0-rc1 (L₀)
  - SPECJBB (2013)
    - Backend only, Controller/Transaction Injectors on a different host
    - Qemu cmdline: -smp n (1, 2, 4 and 8) -m 16G -cpu qemu64
    - Compare L1 and L2 performance numbers
  - Kernel Compilation
    - Use “time” to measure compilation times under the same setup
AMD Performance Evaluation

- Kernel Compilation

![Bar Chart](chart.png)
• Kernel Compilation (Evaluation)
  – Comparable times across the vCPU range
  – “make” is CPU intensive
- **SPECJBB (Distributed with Backend in L$_2$)**

<table>
<thead>
<tr>
<th>Number of vCPUS</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>71</td>
<td>59</td>
</tr>
<tr>
<td>2</td>
<td>61</td>
<td>59</td>
</tr>
<tr>
<td>4</td>
<td>61</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

**AMD Performance Evaluation**
SPECJBB (Evaluation)

- $L_2$ nearly at 50% of $L_1$'s performance
  - TODO: Investigating bottlenecks in the nested setup

- Bottlenecks
  - I/O Bottlenecks? The test setup creates a qcow2 image inside $L_1$
    - File systems are nested
  - Can APIC-v help?
Test Environment

- Host ($L_0$) – IvyTown_EP 16 Cores 128GB RAM
- Qemu options to run $L_1$: `-cpu host -m 20G -smp 10`
- Qemu options $L_1$ uses to run $L_2$: `-cpu qemu64 -m 8G -smp 8`

Guest Status ... not so good news
## Intel - Status

- Some not yet impressive matrix

<table>
<thead>
<tr>
<th>L1 Guest</th>
<th>L2 Guest</th>
<th>RHEL 6.5 64-bit</th>
<th>RHEL 6.5 32-bit</th>
<th>Windows 7 64-bit</th>
<th>Windows 7 32-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xen</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>KVM</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>VMware ESX</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>VMware Player</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>HAXM</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>Win7 XP Mode</td>
<td>N/A</td>
<td>N/A</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>Hyper-V</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
<td>🤖</td>
</tr>
</tbody>
</table>
• Kernel Compilation

Approximate Time

<table>
<thead>
<tr>
<th>Number of vCPUS</th>
<th>L1</th>
<th>L2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>41</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>31</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>32</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
<td>19</td>
</tr>
</tbody>
</table>
• Kernel Compilation (Evaluation)
  - CPU intensive workloads fare quite well
  - But .... do they always?
Intel Performance Evaluation

- **SPECJBB**

![Graph showing max-jOPS (%) for different numbers of vCPUs (L1 and L2)](image)
Intel Performance Evaluation

- **SPECJBB (Evaluation)**
  - What went wrong?
  - Incorrect Test Setup?
  - Newer machines => newer processor features => how is Nested Virtualization affected?
  - Maturity: still needs “right setup” to work

I wish I was better :(
Nested Virtualization and Migration

• Nested VMs implies no migration! ;-)  

But in all seriousness:

• Challenge: Live migrate $L_1$ with all its $L_2$ guests

• Save all nested state: $\text{vmcs}_{12}$, struct nested_vmx, etc

but how?
Nested Virtualization and Migration

• One option:
  – Force an exit from L₂ to L₁ (if running in L₂) – feasible with all L₁ setups?
  – Save all current vmcs₀₂ state to vmcs₁₂
  – L₂ specific dirtied pages need to be copied
  – Nested state metadata gets transferred to destination with L₁'s memory
  – If running in L₂ on source, need to do the same on destination

• Another option:
  – Save/restore additional CPU states, just like additional registers
Nested IOMMU

• Use cases
  – Testing
  – Device assignment to L2
• History
  – AMD IOMMU emulation for QEMU (Eduard-Gabriel Munteanu, 2011)
  – Lacking memory layer abstractions
  – Required many device model hooks
• SPARC QEMU model with own IOMMU layer
Nested IOMMU - Today

- IOMMU support in QEMU memory layer, used for
  - POWER
  - Alpha
  - ...and Intel!
- VT-d emulation developed as GSoC project by Le Tan
  - DMAR emulation, supports all PCI device models
  - Error reporting
  - Cache emulation
- VT-d interrupt remapping emulation
  - Working prototype
  - Lacks error reporting
Nested IOMMU – Open Topics

- Support for physical devices
  - Full in-kernel IOMMU model?
    => ARM SMMU model by Will Deacon, see Linux Plumber IOMMU track
  - Use of VFIO from userspace model?
- IR emulation with in-kernel irqchips
  - Requires extension to translate IOAPIC IRQs
- AMD IOMMU, reloaded?
Wrap-Up

- AMD Nested Virtualization support in good shape
  - Regular **testing** required nevertheless (autotest?)
- Intel Nested Virtualization
  - Add missing mandatory features
  - More **testing** (Intel integration tests 😊, autotest?)
- Once stable, address migration
- IOMMU emulation & nesting approaching
- Non-x86...?