Nested Virtualization on ARM

NEVE: Nested Virtualization Extensions

Jin Tack Lim        Christoffer Dall        Shih-Wei Li
Jason Nieh        Marc Zyngier

jitack@cs.columbia.edu    christoffer.dall@linaro.org    shih-wei@cs.columbia.edu,
nieh@cs.columbia.edu    marc.zyngier@arm.com
Nested Virtualization
Terminology

- **Host Hypervisor**
- **Guest Hypervisor**
- **Nested VM**
- **App**
- **Kernel**

Hardware
Use Cases

1. Run guest operating systems with built-in virtualization.
2. IaaS hosting private clouds
3. Test your hypervisor in a VM
4. Debug your hypervisor in a VM
5. Develop hypervisors using a cloud
ARM Virtualization Extensions

EL0
- User Space

EL1
- Kernel

EL2
- Hypervisor
ARM Nested Virtualization

VM

EL0
User Space

EL1
Kernel

Virtual EL2
Guest Hypervisor

EL2
Host Hypervisor

VM

EL0
User Space

EL1
Kernel

Guest Hypervisor

Guest Hypervisor
ARM Nested Virtualization

- EL0: User Space
- EL1: Kernel
- EL ??: Guest Hypervisor
- EL2: Host Hypervisor
ARMv8.0 Nested Virtualization

EL0
- User Space
- Guest Hypervisor

EL1
- Kernel

EL0
- Guest Hypervisor

EL2
- Host Hypervisor

Trap-and-emulate
ARMv8.0 Nested Virtualization

- **EL0**: User Space
- **EL1**: Kernel
- **EL1**: Guest Hypervisor
- **EL2**: Host Hypervisor

**-and-emulate**
ARMv8.3 Nested Virtualization

- Gives you software emulation of vEL2 in EL1
- HCR_EL2.NV:
  - Traps EL2 operations executed in EL1 to EL2
  - Traps `eret` to EL2
  - CurrentEL reports EL2 even in EL1
KVM/ARM Nested Virtualization Implementation

- EL2 Emulation
- Stage 2 MMU Virtualization
- Hyp Timer Virtualization
- Nested Virtual Interrupts
Nested CPU Virtualization

```c
struct kvm_cpu_context {
    u64 sys_regs[NR_SYS_REGS];
    + u64 el2_regs[NR_EL2_REGS];
}

struct kvm_vcpu_arch {
    ...
    struct kvm_cpu_context ctxt;
}
```
Hypervisor-VM Switch

- **EL0**: App, App
- **EL1**: Linux, KVM
- **EL2**: KVM, Kernel

Save EL1 sys_regs

Restore EL1 sys_regs
Hypervisor-Hypervisor Switch

EL0
App
App

EL1
Linux
KVM

EL2
Kernel
Guest Hypervisor

Save/restore EL1 sys_regs
Save/restore el2_regs
Emulating EL2 in EL1

• Define mapping of EL2 registers to EL1 registers

• Example: TTBR0_EL2 to TTBR0_EL1

• Example: SCTLR_EL2 adapted to SCTLR_EL1

• Shadow EL1 registers
Nested CPU Virtualization

```c
struct kvm_cpu_context {
    u64 sys_regs[NR_SYS_REGS];
    u64 el2_regs[NR_EL2_REGS];
    u64 shaow_sys_regs[NR_SYS_REGS];
};

struct kvm_vcpu_arch {
    ...  
    struct kvm_cpu_context ctxt;
};
```
Shadow Registers

- \&sys\_regs
- \&shadow\_sys\_regs

- PSTATE.mode == EL0/1
- PSTATE.mode == EL2

- u64 *vcpu->ctxtx.hw_regs
Virtual Exceptions

- Trap to virtual EL2
- “Forward” exceptions
- Emulate virtual exceptions
Virtual Exceptions

- Returning from virtual EL2
- Trap `eret` to EL2 (ARMv8.3)
- Emulate virtual exception return
KVM/ARM Nested Virtualization Implementation

- EL2 Emulation
- Stage 2 MMU Virtualization
- Hyp Timer Virtualization
- Nested Virtual Interrupts
Memory Virtualization

Stage 1: VA -> IPA
Memory Virtualization

Stage 1: VA -> IPA
Stage 2: IPA -> PA
Memory Virtualization

EL0
User Space

EL1
Kernel

EL2
Guest Hypervisor

Stage 1: VA -> IPA

Stage 2: IPA -> PA
Memory Virtualization

Stage 1: VA -> IPA

Virtual stage 2

Shadow Stage 2: IPA -> PA
KVM/ARM Nested Virtualization Implementation

- EL2 Emulation
- Stage 2 MMU Virtualization
- Hyp Timer Virtualization
- Nested Virtual Interrupts
Nested Timer Virtualization

- ARM provides a virtual and physical timer in EL1
- EL2 provides a separate EL2 “hyp” timer
- Nested KVM/ARM supports a virtual CPU with EL2 and the hyp timer
KVM/ARM Nested Virtualization Implementation

- EL2 Emulation
- Stage 2 MMU Virtualization
- Hyp Timer Virtualization
- Nested Virtual Interrupts
ARM Generic Interrupt Controller (GIC)
ARM Generic Interrupt Controller (GIC)
Nested Interrupt Virtualization

- Deliver virtual interrupts from the host to the VM
Nested Interrupt Virtualization

- Deliver virtual interrupts from the guest hypervisor to the nested VM
- Shadow list registers
- The nested VM can ACK and EOI virtual interrupts without trapping
Performance Evaluation

- Problem: No ARMv8.3 hardware available.
- Solution: Use ARMv8.0 hardware with the software modification
Emulating v8.3 on v8.0

VM

Nested VM

App
App
App

EL0

EL1

OS Kernel

Guest Hypervisor

HVC
HVC
HVC
HVC

Paravirtualization

EL2

Host Hypervisor

ARMv8.0 Hardware
Hypercall MicroBenchmark

EL0

App  App  App

EL1

OS Kernel

Hypercall  Return

EL2

Hypervisor

EL0

App  App  App

EL1

OS Kernel

Hypercall  Guest Hypervisor  Return

EL2

Host Hypervisor

VM

Nested VM
Hypercall MicroBenchmark

<table>
<thead>
<tr>
<th>ARMv8.3</th>
<th>VM</th>
<th>Nested VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle counts</td>
<td>2,729</td>
<td>422,720</td>
</tr>
<tr>
<td>Ratio to VM</td>
<td>1</td>
<td>155x</td>
</tr>
</tbody>
</table>
Application Benchmarks

Normalized overhead (lower is better)

ARMv8.3 VM
ARMv8.3 Nested

Kernbench
Hackbench
SPECjvm2008
TCP RR
TCP STREAM
TCP MAERTS
Apache
Nginx
Memcached
MySQL
Nested VM Exit/Entry on ARM

> 120 traps
NEVE: NEsted Virtualization Extensions for ARM

- Supports unmodified guest hypervisors and OSes
- Improves performance by providing register redirection
Register Classification

- VM registers: EL1 registers only affecting the nested VM’s execution
- Hypervisor registers: EL2 registers affecting the hypervisor’s execution
VM Registers

This is when VM register states are used
VM Registers: Logging to Memory

Without NEVE

`msr x0, TTBR0_EL1`

Trap!

Memory
VM Registers: Logging to Memory

msr x0, TTBR0_EL1

With NEVE

Memory

TTBR0_EL1
Hypervisor control registers

- Can’t apply the technique for VM registers
- They have an immediate impact (EL2 system registers)
- Traps are handled by redirecting to EL1 registers in software
Hypervisor control registers

- Can’t apply the technique for VM registers
- They have an immediate impact (EL2 system registers)
- Traps are handled by redirecting to EL1 registers in software
- Redirect in hardware instead!
# Hypercall MicroBenchmark

<table>
<thead>
<tr>
<th></th>
<th>ARMv8.3</th>
<th>NEVE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VM</td>
<td>Nested VM</td>
</tr>
<tr>
<td>Cycle counts</td>
<td>2,729</td>
<td>422,720</td>
</tr>
<tr>
<td>Ratio to VM</td>
<td></td>
<td>155x</td>
</tr>
<tr>
<td>Trap counts</td>
<td>1</td>
<td>126</td>
</tr>
</tbody>
</table>
# Application Workloads

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernbench</td>
<td>Kernel compile</td>
<td>Netperf TCP_RR</td>
<td>Network performance</td>
</tr>
<tr>
<td>Hackbench</td>
<td>Scheduler stress</td>
<td>Netperf TCP STREAM</td>
<td>Network performance</td>
</tr>
<tr>
<td>SPECjvm2008</td>
<td>Java Runtime</td>
<td>Netperf TCP MAERTS</td>
<td>Network performance</td>
</tr>
<tr>
<td>MySQL</td>
<td>Database management</td>
<td>Apache</td>
<td>Web server stress</td>
</tr>
<tr>
<td>Memcached</td>
<td>Key-Value store</td>
<td>Nginx</td>
<td>Web server stress</td>
</tr>
</tbody>
</table>
Experimental Setup

- **ARM Hardware**
  - APM X-Gene (ARMv8.0)
  - 8-way SMP
  - 64 GB RAM

- **x86 Hardware**
  - Intel E5-2630 v3
  - VMCS Shadowing
  - 8-way SMP
  - 128 GB RAM

- **Native/VM/Nested VM**
  - 4-way SMP
  - 12 GB RAM
  - Virt I/O (VM/nested VM)
  - 10 Gb Ethernet

- **Software**
  - KVM on KVM
  - v4.10
Application Benchmarks

Normalized overhead
(lower is better)

ARMv8.3 VM
ARMv8.3 Nested
NEVE Nested

Kernbench
Hackbench
SPECjvm2008
TCP RR
TCP STREAM
TCP MAERTS
Apache
Nginx
Memcached
MySQL
Application Benchmarks

Normalized overhead (lower is better)

- ARMv8.3 VM
- ARMv8.3 Nested
- NEVE Nested
- x86 Nested VM

Benchmarks:
- Kernbench
- Hackbench
- SPECjvm2008
- TCP RR
- TCP STREAM
- TCP MAERTS
- Apache
- Nginx
- Memcached
- MySQL

Linaro
Conclusion

- We have an implementation of KVM/ARM for v8.3
- Evaluated nested virtualization performance by emulating ARMv8.3
- Nested virtualization on ARMv8.3 incurs high overhead
  - Due to the exit multiplication problem
- NEVE enhances performance significantly by reducing number of traps
- NEVE is used as basis for extended nested virtualization support in ARMv8.4
- NEVE to appear at SOSP later month - read the paper for more details
Code

- Nested CPU Virtualization patches for ARMv8.3 [RFC v2]:

- Nested Memory Virtualization patches for ARMv8.3 [RFC]:
  https://lists.cs.columbia.edu/pipermail/kvmarm/2017-October/027286.html

- v8.3 and NEVE Paravirtualization on Linux v4.12-rc1:
  https://github.com/columbia/nesting-pub

- QEMU Patches:
  https://github.com/columbia/qemu-pub nested-v2.3.0-model