Optimizing the Design and Implementation of the Linux ARM Hypervisor

Christoffer Dall, Shih-Wei Li, Jason Nieh

Computer Science Department

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK
Virtualization

- Essential technology for servers and cloud computing
- Support running multiple virtual machines on a piece of hardware
- Efficient partitioning and utilization of hardware resources
Virtualization on ARM

ARM Servers
Virtualization

Native

Virtual Machines

App  App  App

OS Kernel

Hardware

VM
App  App
Kernel

VM
App  App
Kernel

Hypervisor

Hardware
Virtualization
Hypervisor Design

Type 1 (Standalone)

VM
- App
- App
- Kernel

VM
- App
- App
- Kernel

Hypervisor

Hardware
Hypervisor Design

Type 1 (Standalone)

- VM
  - App App
  - Kernel

Hypervisor

Hardware

Type 2 (Hosted)

- OS Kernel
- Hypervisor

App

App App

Kernel

Hardware
Hypervisor Design

Type 1 (Standalone)

Dom0

VM
App
App

Kernel

Hypervisor

Hardware

Xen

Type 2 (Hosted)

VM
App
App

Kernel

App

OS Kernel

Hypervisor

Hardware

KVM
Hypervisor OS Kernel

- A full OS to support hypervisor functionality
Overview

• Cost in transitions from the VM to the Hypervisor OS Kernel is critical to virtualization performance

• Optimize the design and performance of KVM/ARM
  • Take advantage of unique features in ARM Virtualization support
  • Reduce the cost of running Hypervisor OS Kernel
Hardware virtualization support

• Modern hypervisors leverage hardware virtualization support in their design

• Different hardware features provided by the architectures

• Different approach to run Hypervisor OS Kernel
x86 Virtualization Support (Intel VMX)

Root (Hypervisor)

Non-Root (VM)
KVM x86

Ring 3

Root

Host Userspace

Hypervisor OS (Linux)

Ring 0

Non-Root

VM

VM Userspace

VM Kernel
ARM Virtualization Extensions

- **EL0**: User
- **EL1**: Kernel
- **EL2**: Hypervisor
KVM/ARM

EL0

Host Userspace

EL1

Hypervisor OS (Linux)  KVM

VM

VM Userspace

VM Kernel

EL2

KVM lowvisor
Sharing kernel mode state

• Multiplexing kernel mode is expensive!

• Multiplexing the shared kernel mode in software is more expensive than hardware
Separate mode for Hypervisor OS kernel

EL0
Host Userspace

EL1

EL2
Hypervisor OS (Linux) KVM

VM
VM Userspace

VM Kernel
Challenges

- EL2:
  - Separate CPU mode
  - Different page table formats
  - Limited capabilities compared to EL1
Challenges

• EL2:
  • Separate CPU mode
  • Different page table formats
  • Limited capabilities compared to EL1
  • Cannot run unmodified OS in EL2!
Running Linux and KVM in EL2

• Hardware: VHE (Virtualization Host Extensions)
  • Hardware is not available yet!
Running Linux and KVM in EL2

- Hardware: VHE (Virtualization Host Extensions)
  - Hardware is not available yet!

- Software: Change Linux to run in EL2
KVM/ARM Redesign

• Avoid multiplexing kernel mode states

• Avoid operating across different CPU modes
  • Allow the Hypervisor OS kernel to program the EL2 hardware

• Reduce code complexity
Experimental Setup

**ARM Hardware**
- AMD Seattle Rev B0
- 2 GHz AMD A1100 CPU
- 8-way SMP
- 16 GB RAM
- 10 GB Ethernet

**x86 Hardware**
- Dell PowerEdge r320
- 2.1 GHz Intel Xeon E5-2450
- 8-way SMP
- 16 GB RAM
- 10 GB Ethernet

**VM**
- 4-way SMP
- 12GB RAM
- Passthrough NIC
Hypercall benchmark

- No-op Hypercall: VM traps to the hypervisor and returns immediately
- Base cost for hypervisor to serve the virtual machines
- Include the cost of transitioning between the VM and the hypervisor
  OS kernel in KVM
Hypercall benchmark

<table>
<thead>
<tr>
<th>CPU cycles</th>
<th>ARM</th>
<th>ARM EL2 OPT</th>
<th>x86</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypercall</td>
<td>6,413</td>
<td>752</td>
<td>1,437</td>
</tr>
</tbody>
</table>
## Application Workloads

<table>
<thead>
<tr>
<th>Applications</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernbench</td>
<td>Kernel compile</td>
</tr>
<tr>
<td>Hackbench</td>
<td>Scheduler stress</td>
</tr>
<tr>
<td>Netperf (TCP_RR)</td>
<td>Network performance</td>
</tr>
<tr>
<td>Apache</td>
<td>Web server stress</td>
</tr>
<tr>
<td>Memcached</td>
<td>Key-Value store</td>
</tr>
</tbody>
</table>
Application Results

![Application Results Chart]

- **Kernbench**
- **Hackbench**
- **Netperf**
- **Apache**
- **Memcached**

The chart compares normalized performance across different benchmarks and platforms:

- **ARM**
- **ARM EL2 OPT**
- **x86**
Conclusion

• Optimized the design and performance of KVM/ARM
  • Improved its performance by roughly an order of magnitude
  • Accomplished faster performance than KVM on x86
• Benefit from future ARM hardware with VHE support
• Many of the proposed changes have been merged to mainline Linux
Q&A