NOVA Microhypervisor on ARMv8-A
FOSDEM 2020

Udo Steinberg
BedRock Systems, Inc.

February 2, 2020
Outline

1. NOVA Microhypervisor
2. ARMv8-A Virtualization
3. Current Status, Demo, Roadmap
The microhypervisor is the **only** privileged component.  
Every virtual machine has its own VMM instance.

---

1 Ignoring SMM and Firmware, which are beyond our control.
The microhypervisor is the *only* privileged component \(^1\)

Every virtual machine has its own VMM instance

---

\(^1\) Ignoring TF-A, Monitor and TEE, which are beyond our control
- Capability is pointer to KObject or PFrame + permissions
- Protection Domain has Object Space, Memory Space, ...
- Hypercall `ctrl pd` with take/grant semantics replaces MDB
Protection Domains, Execution+Scheduling Contexts, Portals

Semaphores for Synchronization and Interrupt Delivery

Hypercall interface uses capabilities for all operations

Synchronous IPC with timeslice donation $\Rightarrow$ priority inheritance

MTD defines number of words to copy $\text{UTCB}_{\text{caller}} \leftrightarrow \text{UTCB}_{\text{callee}}$
vCPU state saved to / restored from VMCB
- Microhypervisor synthesizes IPC call on behalf of vCPU
- Destination portal selected based on type of event
- IPC reply from VMM provides updated architectural state
- $\text{MTD}_{\text{ARCH}}$ defines state to copy VMCB $\leftrightarrow$ UTCB$_{\text{handler}}$
**ARMv8-A** Message Transfer Descriptor (MTD\_ARCH)

<table>
<thead>
<tr>
<th></th>
<th>GIC</th>
<th>TMR</th>
<th>EL2.HCR</th>
<th>EL2.HPFAR</th>
<th>EL2.ESR.FAR</th>
<th>EL2.TDR</th>
<th>EL1.SCLTR</th>
<th>EL1.VBAR</th>
<th>EL1.MAIR</th>
<th>EL1.TCR</th>
<th>EL1.TTBR</th>
<th>EL1.AFSR</th>
<th>EL1.ESR.FAR</th>
<th>EL1.ESR.SPSR</th>
<th>EL1.TDR</th>
<th>EL1.SP</th>
<th>A32.DACR.IFSR</th>
<th>A32.SPSR</th>
<th>EL0.TDR</th>
<th>EL0.SP</th>
<th>FPR</th>
<th>GPR</th>
<th>POISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>31, 30</td>
<td>24, 23</td>
<td>22, 21</td>
<td>20, 19</td>
<td>18, 17</td>
<td>16, 15</td>
<td>14, 13</td>
<td>12, 11</td>
<td>10, 9</td>
<td>8, 7</td>
<td>4, 3</td>
<td>2, 1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**x86** Message Transfer Descriptor (MTD\_ARCH)

<table>
<thead>
<tr>
<th></th>
<th>FPU</th>
<th>TSC</th>
<th>STA</th>
<th>INJ</th>
<th>CTRL</th>
<th>QUAL</th>
<th>LBR</th>
<th>DR</th>
<th>CR</th>
<th>IDTR</th>
<th>GDTTR</th>
<th>LDTTR</th>
<th>TR</th>
<th>CS/SS</th>
<th>FS/GS</th>
<th>DS/ES</th>
<th>FLAGS</th>
<th>IP</th>
<th>GPR0-15</th>
<th>GPR4-7</th>
<th>GPR0-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>19, 18</td>
<td>17, 16</td>
<td>15, 14</td>
<td>13, 12</td>
<td>11, 10</td>
<td>9, 8</td>
<td>7, 6</td>
<td>5, 4</td>
<td>3, 2</td>
<td>1, 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For every bit set to 1, the corresponding architectural state is transmitted from the vCPU to the VMM handler or vice versa.
Hypervisor **context-switches** FPU state (32x128bit SIMD registers) between ECs lazily

FPU Access Disabling/Enabling
- Switch away from FPU owner ⇒ disable FPU
- Switch back to FPU owner ⇒ enable FPU

FPU switch moved out of critical IPC path using hazard tricks

<table>
<thead>
<tr>
<th>CPU Hazard Bit</th>
<th>EC Hazard Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPU is disabled (0)</td>
<td>EC is not FPU owner (0)</td>
</tr>
<tr>
<td>FPU is enabled (1)</td>
<td>EC is FPU owner (1)</td>
</tr>
</tbody>
</table>

Slow path taken only if CPU Hazard ⊕ EC Hazard is 1

FPU use must be explicitly declared during EC creation
Unified interrupt injection interface for GICv2/GICv3
Hypercall assign int for configuring and routing SPIs
**Physical Timer**

- Real system counter
- Can be trapped ⇒ **Trap & emulate** timer
- pTimer interrupt emulated with semaphore timeouts ⇒ Asynchronous delivery

**Virtual Timer**

- System counter - offset
- Cannot be trapped ⇒ **Context-switch** timer
- vTimer interrupt temporarily belongs to current VM ⇒ Synchronous via Portal
System MMU protects against rogue DMA

Limited number of stream mapping groups and translation contexts managed by partition manager

Hypercall `assign_dev` for configuring SID/SMG/CTX and binding a device to a protection domain
Currently Supported ARM Platforms

- **Avnet Xilinx Ultra 96**
  - 4x Cortex-A53
  - GICv2
  - SMMUv2

- **NXP i.MX 8MQuad**
  - 4x Cortex-A53
  - GICv3

- **Renesas R-Car M3/H3**
  - 4x Cortex-A53
  - 4x Cortex-A57
  - GICv2

- **Raspberry Pi 4B**
  - 4x Cortex-A72
  - GICv2

- **QEMU Virt Platform**
  - Cortex-A
  - GICv2/GICv3

Udo Steinberg (BedRock Systems, Inc.)

NOVA Microhypervisor on ARMv8-A

February 2, 2020
Current Status: Demo

Guest

- vCPU
- UART
- vGIC
- vETH
- vCPU
- vCPU

Host

- vUART
- vGIC
- vETH
- vUART
- vGIC
- vETH
- vUART
- vGIC
- vETH

- UART Driver
- UART Multiplexer
- Virtual Ethernet Switch

Partition Manager

NOVA Microhypervisor

- UART
- Display
- SD
- USB
- WiFi
- Core
- Core
- Core

Udo Steinberg (BedRock Systems, Inc.)

NOVA Microhypervisor on ARMv8-A

February 2, 2020
Roadmap

- **Architecture Unification**
  - Merge significant portions of the x86 and ARMv8 source code
  - \{src/x86_64, src/aarch64\} \implies src
  - \{inc/x86_64, inc/aarch64\} \implies inc

- **Support for newer ARM features (ARMv8.1 – ARMv8.6)**
- **Additional NOVA functionality**
  - Relocatable microhypervisor binary
  - VM introspection support
  - Improved kernel resource management
  - Useful external features and bug fixes

- **Performance Optimizations**
- **Formal Verification of the NOVA microhypervisor**
  - ... and of components running on top of it
/* 
* \arg{v1} "x" (Vint v1)
* \arg{v2} "y" (Vint v2)
* \pre empSP
* \post{}[Vint (trim 32 (v1+v2))] empSP
*/

auto add_func (uint32 x, uint32 y) {
    return x + y;
}
Source code available under **GPLv2 license** at:
https://github.com/bedrocksystems/NOVA
https://github.com/udosteinberg/NOVA

Checkout the "**arm**" branch.

Further information (papers, links) at:
https://bedrocksystems.com
http://hypervisor.org