# NOVA Microhypervisor on ARMv8-A

FOSDEM 2020

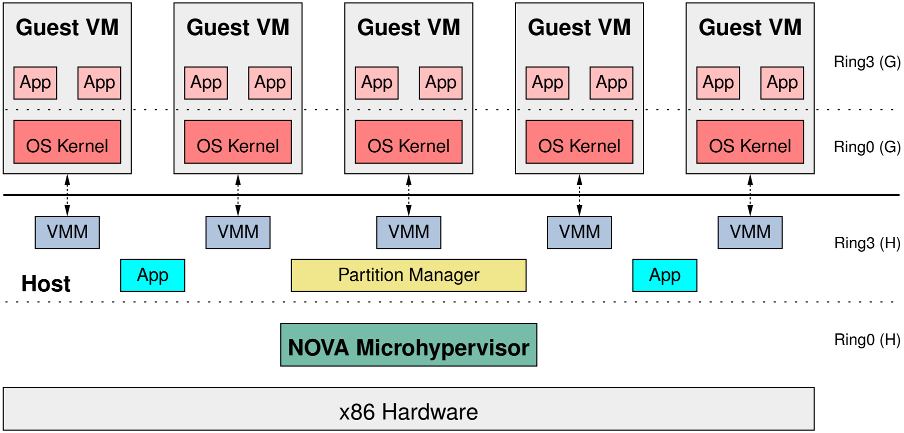
Udo Steinberg

BedRock Systems, Inc.

February 2, 2020

## Outline

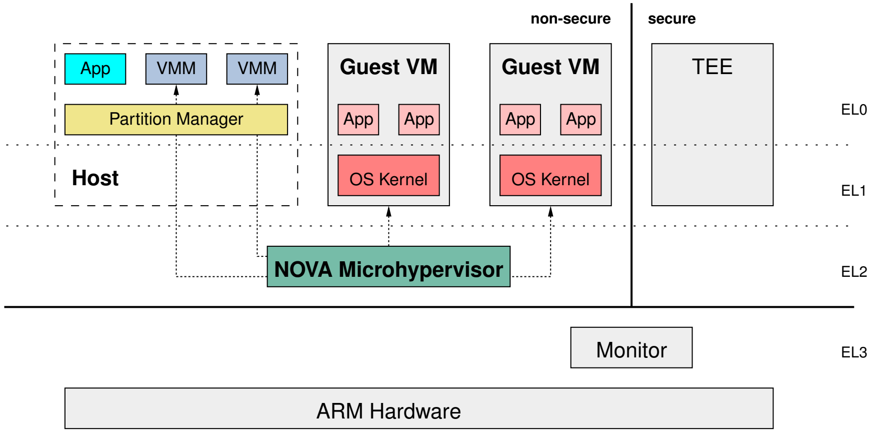
# NOVA: System Architecture – x86



- The microhypervisor is the **only** privileged component [1]
- Every virtual machine has its own VMM instance

[1] Ignoring SMM and Firmware, which are beyond our control

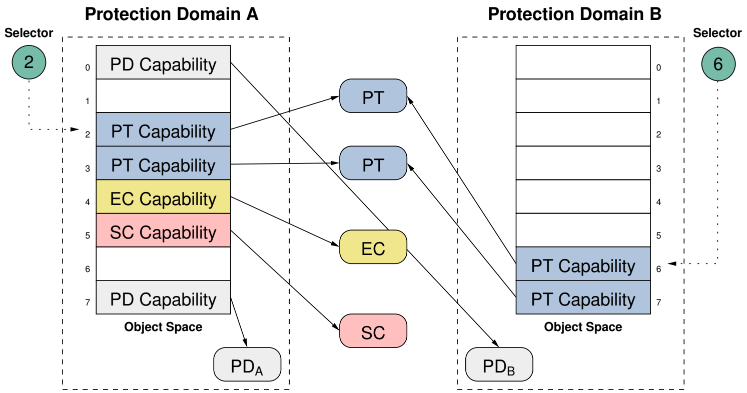
## NOVA: System Architecture – ARMv8-A



- The microhypervisor is the **only** privileged component [1]
- Every virtual machine has its own VMM instance

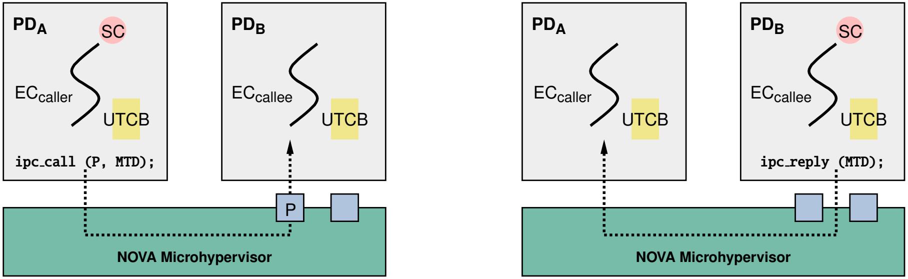
[1] Ignoring TF-A, Monitor and TEE, which are beyond our control

# NOVA: Capability-Based Access Control



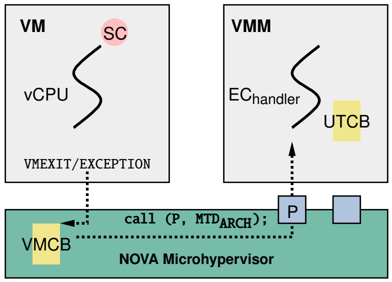
- Capability is pointer to KObject or PFrame + permissions
- Protection Domain has Object Space, Memory Space, ...
- Hypercall `ctrl_pd` with take/grant semantics replaces MDB

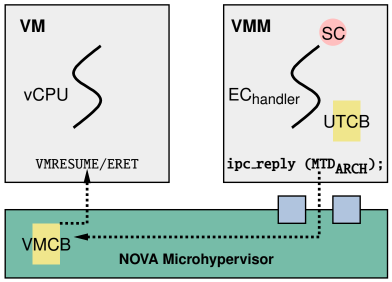
# NOVA: Basic Abstractions



- 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 $\text{UTCB}_{\text{caller}} \rightleftarrows \text{UTCB}_{\text{callee}}$

# NOVA: Handling VM Exits / Exceptions





- 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 $\rightleftarrows$ $\text{UTCB}_{\text{handler}}$

# CPU Virtualization: Architectural State

- **ARMv8-A** Message Transfer Descriptor ($MTD_{ARCH}$)

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GIC | TMR | ▨ | EL2_HCR | EL2_HPFAR | EL2_ESR_FAR | EL2_ELR_SPSR | EL2_IDR | EL1_SCTLR | EL1_VBAR | EL1_MAIR | EL1_TCR | EL1_TTBR | EL1_AFSR | EL1_ESR_FAR | EL1_ELR_SPSR | EL1_IDR | EL1_SP | ▨ | A32_DACR_IFSR | A32_SPSR | ▨ | EL0_IDR | EL0_SP | FPR | GPR | POISON |
| 31 | 30 | | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | | 8 | 7 | | 4 | 3 | 2 | 1 | 0 |

- **x86** Message Transfer Descriptor ($MTD_{ARCH}$)

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FPU | ▨ | TSC | STA | INJ | CTRL | QUAL | LBR | DR | CR | IDTR | GDTR | LDTR | TR | CS/SS | FS/GS | DS/ES | FLAGS | IP | $GPR_{8-15}$ | $GPR_{4-7}$ | $GPR_{0-3}$ |
| 31 | | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

- For every bit set to 1, the corresponding architectural state is transmitted from the vCPU to the VMM handler or vice versa.

# ARM: FPU Virtualization

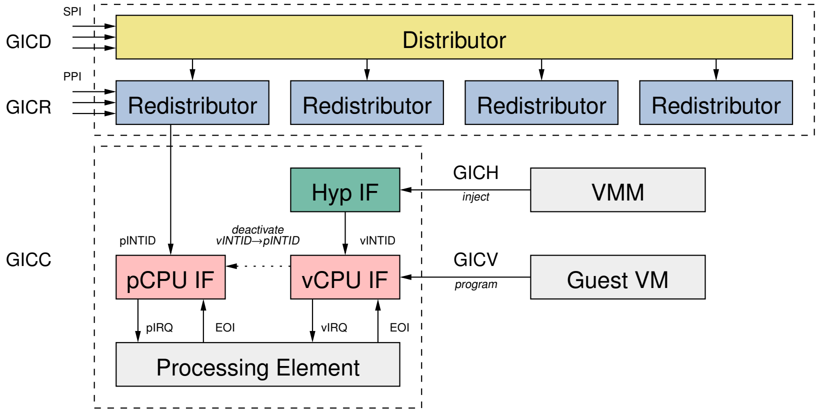
- Hypervisor **context-switches** FPU state (32x128bit SIMD registers) between ECs lazily
- FPU Access Disabling/Enabling
  - Switch away from FPU owner $\Rightarrow$ disable FPU
  - Switch back to FPU owner $\Rightarrow$ enable FPU
- FPU switch moved out of critical IPC path using hazard tricks

| **CPU Hazard Bit** | **EC Hazard Bit** |
|---|---|
| FPU is disabled (0) | EC is not FPU owner (0) |
| FPU is enabled (1) | EC is FPU owner (1) |

Slow path taken only if CPU Hazard $\oplus$ EC Hazard is 1

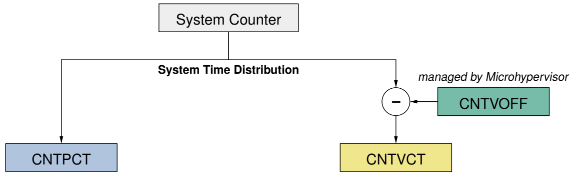
- FPU use must be explicitly declared during EC creation

# ARM: Interrupt Virtualization



- Unified interrupt injection interface for GICv2/GICv3
- Hypercall `assign_int` for configuring and routing SPIs

# ARM: Timer Virtualization



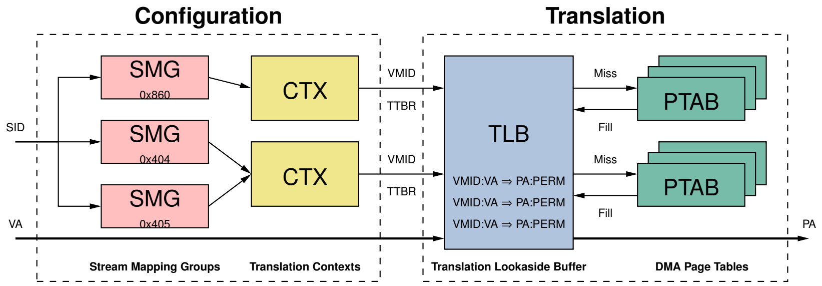
**Physical Timer**

- 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

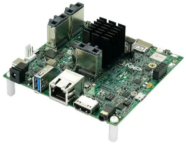
# ARM: System MMU



- 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



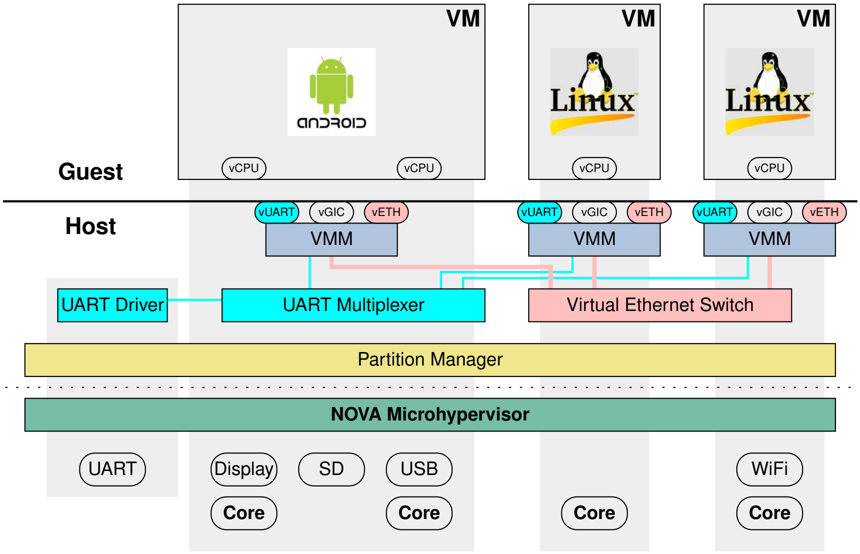
**NXP i.MX 8MQuad**
4x Cortex-A53
GICv3

**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

# Current Status: Demo

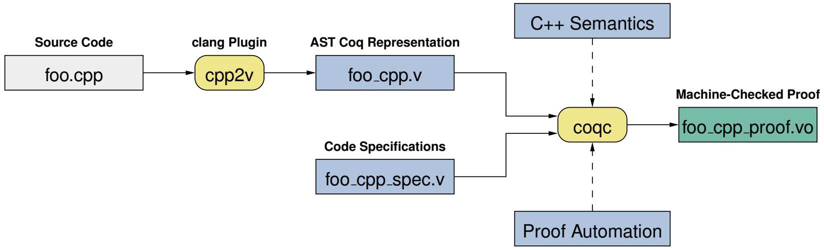


## Roadmap

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

## Formal Verification

```
/*
 * \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`