RISC-V Hypervisor Status

Alistair Francis, Anup Patel, Atish Patra

Linux Plumbers - Lisbon

9th of September 2019
RISC-V H-Extension
RISC-V H-Extension: Spec Status

H-Extension spec close to freeze state

• v0.4-draft was released on June 16th
  – This includes feedback from Open Source virtualisation projects
  – Additions have happened to the spec since
    • htimedelta/htimedeltah CSR (Proposed by WDC – Merged)
    • Dedicated exception causes for Guest page table faults (Proposed by John Hauser – In Review)
    • hgip CSR for better virtual interrupt injection (Proposed by WDC – In Review)
    • htinst & htval2 CSRs for better MMIO emulation (Proposed by WDC and extended by John Hauser – In Review)

• RISC-V Virtualisation is much similar to ARM-VHE then the original AArch64 Virtualisation

• WDC’s initial QEMU, Xvisor and KVM ports were based on v0.3

• They have all been updated to the new v0.4 spec
  – There were limited software changes required between v0.3 and v0.4
    • QEMU required more changes
RISC-V H-Extension: Privilege Mode Changes

New execution modes for guest execution

- HS-mode = S-mode with hypervisor capabilities and new CSRs
- Two additional modes:
  - VS-mode = Virtualized S-mode
  - VU-mode = Virtualized U-mode
RISC-V H-Extension: CSR changes

More control registers for virtualising S-mode

• Additional virtual copies of most S-mode CSRs

• In HS-mode (V=0),
  – “s<xyz>” CSRs point to standard “s<xyz>” CSRs
  – “hs<xyz>” CSRs for hypervisor capabilities
  – “vs<xyz>” CSRs contains VS-mode state

• In VS-mode (V=1)
  – “s<xyz>” CSRs point to virtual “vs<xyz>” CSRs
RISC-V H-Extension: Two-stage MMU

Hardware optimized guest memory management

• Two-Stage MMU for VS/VU-mode:
  – VS-mode page table (Stage1):
    • Translates Guest Virtual Address (GVA) to Guest Physical Address (GPA)
    • Programmed by Guest (same as before)
  – HS-mode guest page table (Stage2):
    • Translates Guest Physical Address (GPA) to Host Physical Address (HPA)
    • Programmed by Hypervisor

• In HS-mode, software can program two page tables:
  – HS-mode page table: Page table to translate hypervisor Virtual Address (VA) to Host Physical Address (HPA)
  – HS-mode guest page table: Same as above

• Format of VS-mode page table, HS-mode guest page table and HS-mode host page table is same (Sv32, Sv39, Sv48, ....)
RISC-V H-Extension: I/O & Interrupts

I/O and guest interrupts virtualization

• Virtual interrupts injected by updating VSIP CSR from HS-mode

• Software and Timer Interrupts:
  – Hypervisor will emulate SBI calls for Guest

• HS-mode guest page table can be used to trap-n-emulate MMIO accesses for:
  – Software emulated PLIC
  – VirtIO devices
  – Other software emulated peripherals
QEMU: Register Swapping

How to handle Hypervisor Register Swapping?

- How to handle the current S-Mode CSR swapping with virtual/hypervisor CSR

- Currently:
  - Using pointers to handle M-Mode CSRs that are exposed as S-Mode (mstatus, mie)
  - Value swapping the S-Mode only CSRs
  - MIP CSR (atomically accessed) is value swapped as well
RISC-V Hypervisors
Which Hypervisors Ported?

- We have ported both Type1 and Type2 hypervisors for RISC-V. This helps us:
  - Provide feedback to RISC-V H-Extension ISA authors
  - Validate functional completeness of the RISC-V H-Extension spec
  - Gives confidence to HW designers for implementing this in HW

- World’s first RISC-V Type1 hypervisor is Xvisor
  (Refer, http://xhypervisor.org/)

- World’s first RISC-V Type2 hypervisor is KVM
  (Refer, https://www.linux-kvm.org/page/Main_Page)
Xvisor RISC-V

- Hypervisor Component
- M-mode Software
- HS-mode Software
- VS-mode Software
- VU-mode Software
- U-mode Software

- Guest User Space
- Guest Kernel
- VirtIO Frontends

- Guest0
- Guest1
- GuestN

- Guest IO Emulation
- CPU Virtualization
- Management Terminal

- Xvisor Hypervisor
- VirtIO Backends
- Orphan vCPUs

- Firmware (OpenSBI)

- VU-mode
- VS-mode
- HS-mode
- M-mode
Linux KVM RISC-V

- **Host User Space**
  - Process1
  - ProcessN
  - Guest User Space
    - Guest Kernel
      - VirtIO Frontends
      - QEMU/KVMTOOL
        (Guest IO Emulation + VirtIO Backends)
  - Process (Guest0)

- **Device Drivers** (Host HW Access)
- **Host Linux Kernel**
- **KVM Module** (CPU Virtualization)
- **Firmware (OpenSBI)**

- **VU-mode**
- **VS-mode**
- **U-mode**
- **HS-mode**
- **M-mode**

**Legend**
- Hypervisor Component
- M-mode Software
- HS-mode Software
- VS-mode Software
- VU-mode Software
- U-mode Software

©2019 Western Digital Corporation or its affiliates. All rights reserved.
KVM RISC-V on GitHub

• KVM RISC-V git repo (shared between Me and Atish):
  https://github.com/kvm-riscv/linux.git

• KVMTOOL RISC-V git repo:
  https://github.com/kvm-riscv/kvmtool.git

• KVM RISC-V wiki:
  https://github.com/kvm-riscv/howto/wiki
  https://github.com/kvm-riscv/howto/wiki/KVM-RISCV64-on-QEMU
Current Status
Upstream Status

• **QEMU:** Hypervisor Extension support patches sent
• **OpenSBI:** Patches sent, waiting for more review comments
• **Xvisor:** Patches merged in Xvisor-next, will be part of next release towards year end
• **Linux KVM:** Patches reviewed and acked, waiting for merge in next Linux release
• **KVMTOOL:** Patches not up-streamed, **we wanted Linux KVM patches to accepted first**
• **QEMU-KVM:** Not started yet, **we wanted Linux KVM patches to accepted first**
• **Libvirt:** Not started yet, this will be done after QEMU-KVM is available
Still To Do

• QEMU
  – Get 32-bit Xvisor working
  – Update implementation with new spec releases
  – Allow changing XLEN for S-mode from M-mode
  – Allow changing XLEN for VS-mode from HS-mode

• Xvisor
  – Get 32-bit Xvisor working
  – Bring-up on real-HW or FPGA
  – Emulate SBI v0.2 and SBI v0.2 extensions for Guest kernel
  – Virtualize vector extensions
  – Allow 32bit Guest on 64bit Host
  – Allow big-endian Guest on little-endian Host and vice-versa
Still To Do

• KVM
  – Get 32-bit KVM working
  – Bring-up on real-HW or FPGA
  – KVM unit test support
  – Emulate SBI v0.2 and SBI v0.2 extensions for Guest kernel
  – Virtualize vector extensions
  – In-kernel PLIC emulation
  – Upstream KVMTOOL changes
  – QEMU KVM support
  – Guest/VM migration support
  – Libvirt support
  – Allow 32bit Guest on 64bit Host
  – Allow big-endian Guest on little-endian Host and vice-versa
Questions & Suggestions