RISC-V Hypervisor Extension
Where are we? What next?

Anup Patel <anup.patel@wdc.com>
Western Digital System Software Research
Linux Plumbers 2020
Outline

• RISC-V H-Extension Status
• KVM RISC-V
• RISC-V Nested Virtualization
• Questions
RISC-V H-Extension Status

H-Extension specifications close to freeze state

• The hypervisor specific ISA in RISC-V is called RISC-V H-Extension

• Key contributors for initial RISC-V H-Extension drafts:
  – Andrew Waterman (SiFive), John Hauser, and Paolo Bonzini (RedHat)

• RISC-V H-Extension draft release history:
  – v0.1-draft was released on 9\textsuperscript{th} November 2017
  – ... few more draft releases ...
  – v0.4-draft was released on 16\textsuperscript{th} June 2019
  – v0.5-draft released on 30\textsuperscript{th} October 2019
  – v0.6-draft released on 8\textsuperscript{th} February 2020
  – v0.6.1-draft released on 5\textsuperscript{th} May 2020

• Most likely v0.6.1-draft is the last draft release

• Western Digital has been co-developing RISC-V H-Extension since v0.4-draft:
  – QEMU (Emulator), Xvisor (Type-1 hypervisor), KVM (Type-2 hypervisor)
KVM RISC-V

The RISC-V port of the KVM hypervisor
KVM RISC-V: High-level View

High-level view of KVM RISC-V

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

Decreasing Privilege Level

Virtualized World

VU
Guest User Space

VS
Guest Linux

HS
Host Linux + KVM

Non-virtualized World

U
QEMU/KVMTOOL

M
Firmware (OpenSBI)

M
HS
U
KVM RISC-V: Current State

What have we achieved so far?

- Key Aspects:
  - Supports H-Extension v0.6.1 draft specification
  - No RISC-V specific KVM IOCTL
  - Supports both RV32 and RV64 Hosts
  - Minimal world-switch and full world-switch via vcpu_load()/vcpu_put()
  - Floating point unit lazy save/restore
  - KVM ONE_REG interface for user-space
  - Timer and IPI emulation in kernel-space
  - PLIC emulation is done in user-space
  - Hugepage support
  - SBI v0.2 interface for Guests
  - Unhandled SBI calls forwarded to KVM userspace
  - Vhost support using ioeventfd

New addition compared to LPC2019
KVM RISC-V: Patches

Where are the patches?

• The state of KVM RISC-V patches:
  – First version of KVM RISC-V series was posted on July 29th 2019
  – Most of the patches were Reviewed-n-Acked in v6 of KVM RISC-V series
  – Currently, we are at v13 of KVM RISC-V series which was sent on July 10th 2020

• Patches blocked on KVM RISC-V patches:
  – KVM RISC-V vhost support using ioeventfd (RFC v1 sent on July 24th 2020)
  – KVM RISC-V SBI v0.2 support (RFC v1 sent on August 3rd 2020)
  – KVMTOOL patches (RFC v4 sent on July 10th 2020)
  – QEMU KVM patches (RFC v2 sent on April 11th 2020)

• Important Links:
  – https://github.com/kvm-riscv/linux.git (KVM RISC-V repository)
  – http://lists.infradead.org/mailman/listinfo/kvm-riscv (mailing List)
KVM RISC-V: To-Do List

What’s next?

• Stage 2 dirty page logging (**work already in-progress**)
• Nested virtualization (**work already in-progress**)
• Trace points
• KVM unit test support
• Virtualize vector extensions
• Guest/VM migration support
• Allow 32bit Guests on 64bit Hosts (**defined in RISC-V spec**)
• Allow big-endian Guests on little-endian Hosts and vice-versa (**defined in RISC-V spec**)
• Anything else?
RISC-V Nested Virtualization

Nested virtualization using RISC-V H-extension
RISC-V Nested: High-level View

Software layers involved in RISC-V Nested Virtualization

- Nested Virtualized World
  - Guest User Space
  - Guest Linux
  - Guest Hypervisor

- Non-virtualized World
  - Host User Space

Decreasing Privilege Level:
- M-mode Software
- HS-mode Software
- VS-mode Software
- VU-mode Software
- U-mode Software
RISC-V Nested: Hypervisor CSR and Instructions

Virtualizing hypervisor CSRs and instructions for Guest Hypervisor

- H<xyz> and VS<xyz> CSRs are **Hypervisor CSRs**
- HFENCE and HLV/HSV instructions are **Hypervisor Instructions**
- Host hypervisor (HS-mode) will handle hypervisor CSR/instruction traps as follows:
  - **Trap from virtual-HS-mode**: Emulate hypervisor CSR (or Instruction) for Guest Hypervisor
  - **Trap from virtual-VS/VU-mode**: Redirect hypervisor CSR (or Instruction) trap to Guest Hypervisor
  - **Trap from virtual-U-mode**: Redirect hypervisor CSR (or Instruction) trap to Guest Hypervisor as illegal instruction trap
RISC-V Nested: WFI and ECALL Instructions

Virtualizing WFI and ECALL instructions for Guest Hypervisor

- Host hypervisor (HS-mode) will handle WFI traps as follows:
  - Trap from virtual-HS-mode: Emulate WFI for Guest Hypervisor
  - Trap from virtual-VS-mode with virtual-HSTATUS.TW == 1: Redirect WFI trap to Guest Hypervisor
  - Trap from virtual-VS-mode with virtual-HSTATUS.TW == 0: Skip WFI instruction
  - Trap from virtual-VU-mode: Redirect WFI trap to Guest Hypervisor
  - Trap from virtual-U-mode: Redirect WFI trap to Guest Hypervisor as Illegal instruction trap

- Host hypervisor (HS-mode) will handle supervisor ECALL traps as follows:
  - Trap from virtual-HS-mode: Treat it as SBI call from Guest Hypervisor
  - Trap from virtual-VS-mode: Redirect supervisor ECALL trap to Guest Hypervisor
RISC-V Nested: Shadow G-Stage

Emulating G-Stage Translation for Guest Hypervisor

• Host Hypervisor need two G-Stage Page Tables:
  – Normal G-Stage page table
    • Translates Guest Physical Address (GPA) for virtual-HS/U-mode
    • One Normal G-Stage page table for Guest Hypervisor
  – Shadow G-Stage page table
    • Translates Guest Physical Address (GPA) for virtual-VS/VU-mode
    • Separate Shadow G-Stage page table for each VCPU of Guest Hypervisor
    • Software walk of Guest Hypervisor G-Stage required for mappings in Shadow G-Stage
    • Inject Guest page faults to Guest Hypervisor when PTE is missing in Guest Hypervisor G-Stage
    • Guest Hypervisor HFENCE instruction will remove mappings from Shadow G-Stage

• Normal and Shadow G-Stage share same VMID assigned for Guest/VM
RISC-V Nested: Nested World-Switch

Additional world-switch for achieving nested virtualization

- Nested world-switch required when switching between Guest Hypervisor *(virtual-HS-mode)* and Guest Linux *(virtual-VS/VU-mode)*

- Guest Hypervisor *(virtual-HS-mode)* to Guest Linux *(virtual-VS/VU-mode)* done when:
  - SRET instruction executed by Guest Hypervisor with virtual-HSTATUS.SPV == 1

- Guest Linux *(virtual-VS/VU-mode)* to Guest Hypervisor *(virtual-HS-mode)* done when:
  - injecting interrupt to Guest Hypervisor
  - Injecting Guest page faults to Guest Hypervisor
  - Redirecting/injecting virtual instruction trap to Guest Hypervisor
Questions ?
Backup
RISC-V H-Extension

The RISC-V Hypervisor Extension
RISC-V H-Extension: Privilege Mode Changes

New privilege modes for guest execution

- Suitable for both Type-1 (Baremetal) and Type-2 (Hosted) hypervisors

- HS-mode for Hypervisor
  - S-mode with hypervisor capabilities

- Two additional modes for Guest:
  - VS-mode = Virtualized S-mode
  - VU-mode = Virtualized U-mode

- In HS-mode (V=0)
  - “s<xyz>” CSRs point to standard “s<xyz>” CSRs
  - “h<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 optimised guest memory management

- One-Stage MMU for HS/U-mode
  - **HS-mode page table (HS-Stage)**
    - Translate hypervisor Virtual Address (VA) to Host Physical Address (HPA)
    - Programmed by Hypervisor using **satp CSR**
  - HS-mode guest page table (G-Stage)
    - Translates Guest Physical Address (GPA) to Host Physical Address (HPA)
    - Programmed by Hypervisor using **hgatp CSR**

- Two-Stage MMU for VS/VU-mode
  - **VS-mode page table (VS-Stage)**
    - Translates Guest Virtual Address (GVA) to Guest Physical Address (GPA)
    - Programmed by Guest using **satp (aka vsatp) CSR**
  - **HS-mode guest page table (G-Stage)**
    - Translates Guest Physical Address (GPA) to Host Physical Address (HPA)
    - Programmed by Hypervisor using **hgatp CSR**

- Format of all above page tables is same
# RISC-V H-Extension: Hypervisor CSRs

More control registers for virtualising S-mode

<table>
<thead>
<tr>
<th>HS-mode CSRs for hypervisor capabilities</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>hstatus</td>
<td>Hypervisor Status</td>
</tr>
<tr>
<td>hideleg</td>
<td>Hypervisor Interrupt Delegate</td>
</tr>
<tr>
<td>hdeleg</td>
<td>Hypervisor Trap/Exception Delegate</td>
</tr>
<tr>
<td>hie</td>
<td>Hypervisor Interrupt Enable</td>
</tr>
<tr>
<td>hgeie</td>
<td>Hypervisor Guest External Interrupt Enable</td>
</tr>
<tr>
<td>htimedelta</td>
<td>Hypervisor Guest Time Delta</td>
</tr>
<tr>
<td>hcounteren</td>
<td>Hypervisor Counter Enable</td>
</tr>
<tr>
<td>hval</td>
<td>Hypervisor Trap Value</td>
</tr>
<tr>
<td>htinst</td>
<td>Hypervisor Trap Instruction</td>
</tr>
<tr>
<td>hip</td>
<td>Hypervisor Interrupt Pending</td>
</tr>
<tr>
<td>hvip</td>
<td>Hypervisor Virtual Interrupt Pending</td>
</tr>
<tr>
<td>hgeip</td>
<td>Hypervisor Guest External Interrupt Pending</td>
</tr>
<tr>
<td>hgatp</td>
<td>Hypervisor Guest Address Translation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HS-mode CSRs for accessing Guest/VM state</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>vsstatus</td>
<td>Guest/VM Status</td>
</tr>
<tr>
<td>vsie</td>
<td>Guest/VM Interrupt Enable</td>
</tr>
<tr>
<td>vsip</td>
<td>Guest/VM Interrupt Pending</td>
</tr>
<tr>
<td>vstvec</td>
<td>Guest/VM Trap Handler Base</td>
</tr>
<tr>
<td>vsepc</td>
<td>Guest/VM Trap Program Counter</td>
</tr>
<tr>
<td>vscause</td>
<td>Guest/VM Trap Cause</td>
</tr>
<tr>
<td>vstval</td>
<td>Guest/VM Trap Value</td>
</tr>
<tr>
<td>vsatp</td>
<td>Guest/VM Address Translation</td>
</tr>
<tr>
<td>vsscratch</td>
<td>Guest/VM Scratch</td>
</tr>
</tbody>
</table>

- New CSRs based on Western Digital feedback
RISC-V H-Extension: MMIO & Interrupts

Guest MMIO and Interrupts virtualization

- Guest virtual interrupts are injected by updating hvip CSR from HS-mode
  - hvip.VSEIP bit for Hypervisor injected virtual external interrupt
  - hvip.VSTIP bit for Hypervisor injected virtual timer interrupt
  - hvip.VSSIP bit for Hypervisor injected virtual inter-processor interrupt

- Virtual timer and inter-processor interrupts injected based on SBI calls from Guest

- Hypervisor can trap-n-emulate Guest MMIO using HS-mode guest page table
  - Software emulated PLIC
  - VirtIO devices
  - Other software emulated peripherals
RISC-V H-Extension: Future Work

What next in RISC-V specifications for virtualization?

• RISC-V H-Extension specification:
  – Optional acceleration for nested virtualization
  – Optional acceleration for G-stage dirty page tracking

• RISC-V SBI specification:
  – SBI extension for guest time scaling
  – SBI extension for para-virt steal time accounting

• New RISC-V interrupt controller specification with virtualization support

• New RISC-V IOMMU specification

• Any thing else ??