VENTANA MICRO

RISC-V irqbypass

Andrew Jones <ajones@ventanamicro.com>

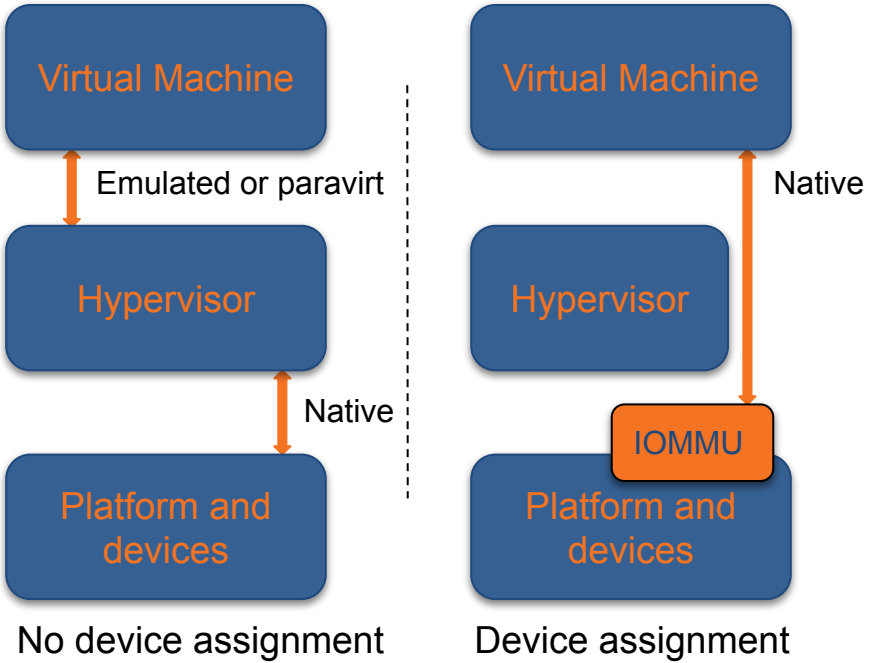
Outline

VENTANA MICRO
RISC-V Performance Leader

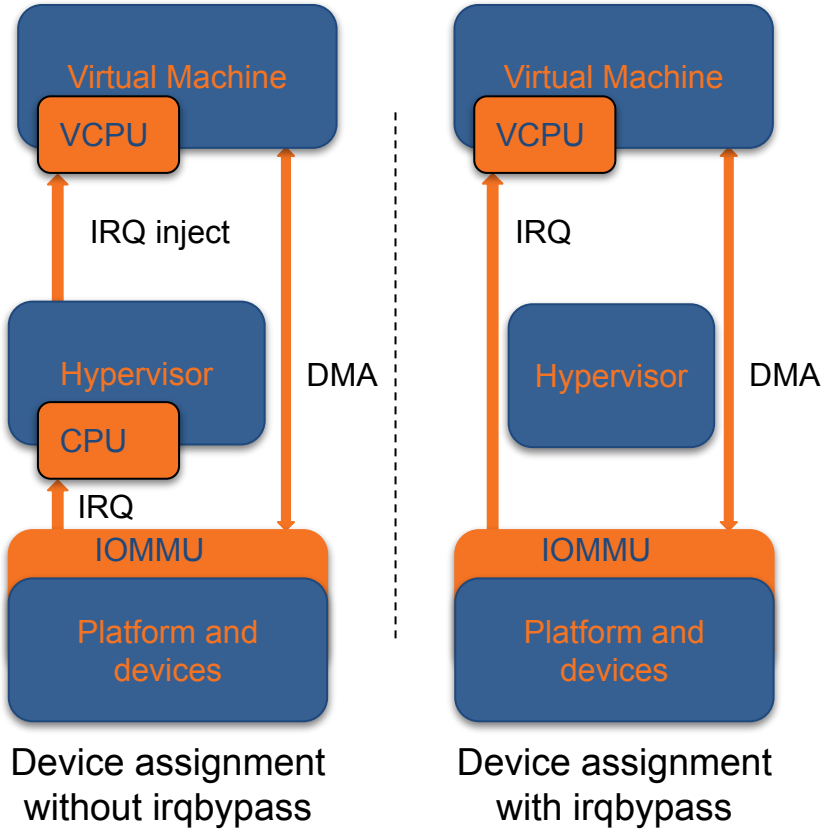
- Device assignment and irqbypass
- RISC-V architecture support
- Linux and KVM support
- TODO

Device assignment and irqbypass

Device assignment



irqbypass

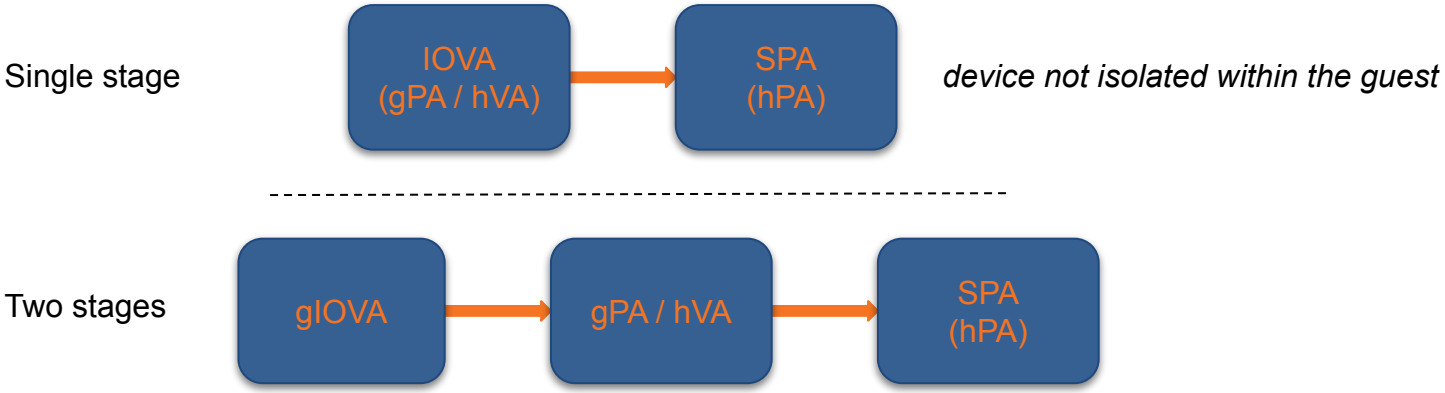

Virtual Machine
VCPU
IRQ inject
Hypervisor
CPU
IRQ
IOMMU
Platform and devices
DMA
Device assignment without irqbypass
Virtual Machine
VCPU
IRQ
Hypervisor
IOMMU
Platform and devices
DMA
Device assignment with irqbypass

RISC-V architecture support

The IOMMU's role

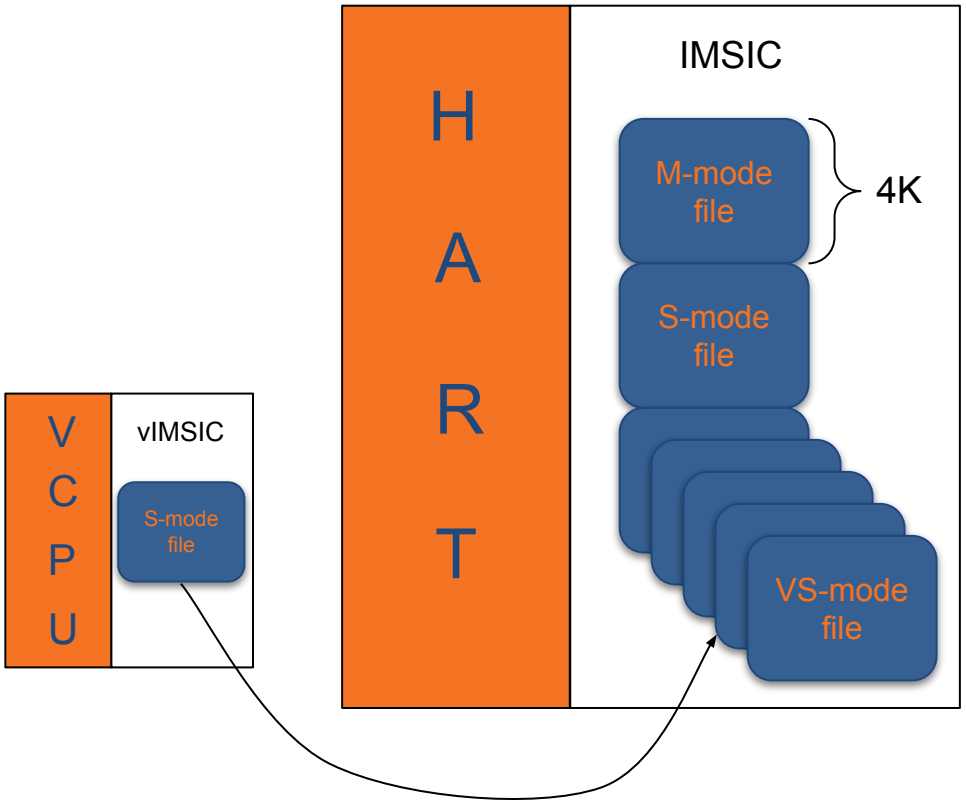
VENTANA MICRO
RISC-V Performance Leader

- DMA
  - Maps device virtual addresses (IOVA) to host physical addresses (hPA)
  - Some IOMMUs, including RISC-V, support two stage translation



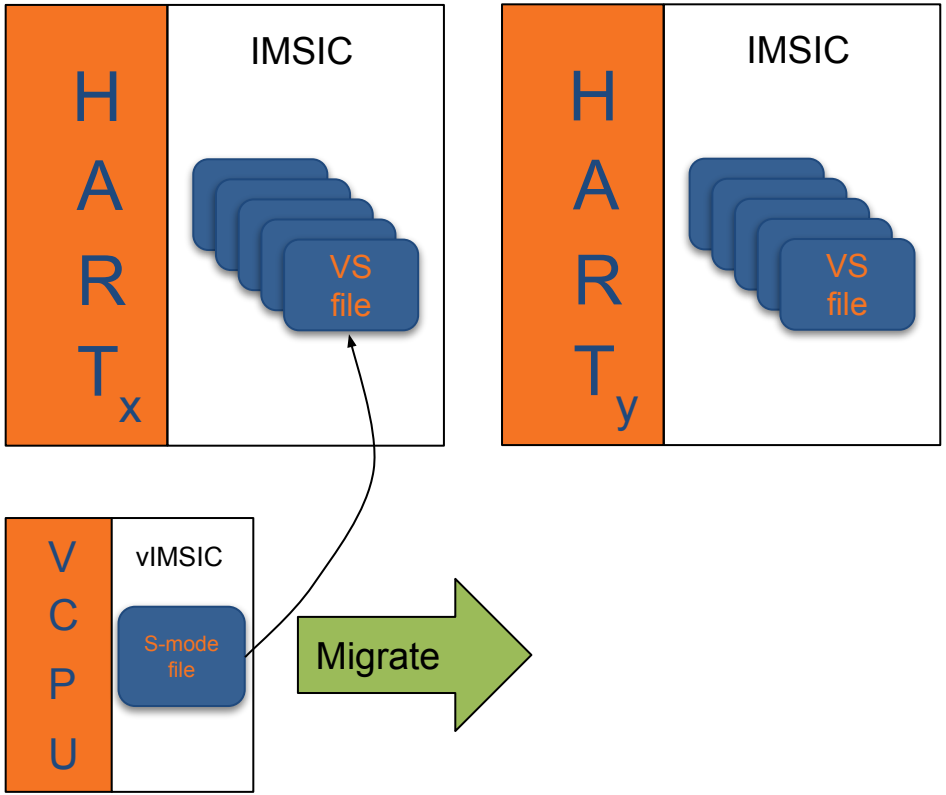
- IRQ
  - Despite an MSI just being a write to an address, the RISC-V IOMMU does not use the same translation configuration as it does for DMA
    ...To see why, we'll take a look at RISC-V guest MSI support...

Guest MSI support



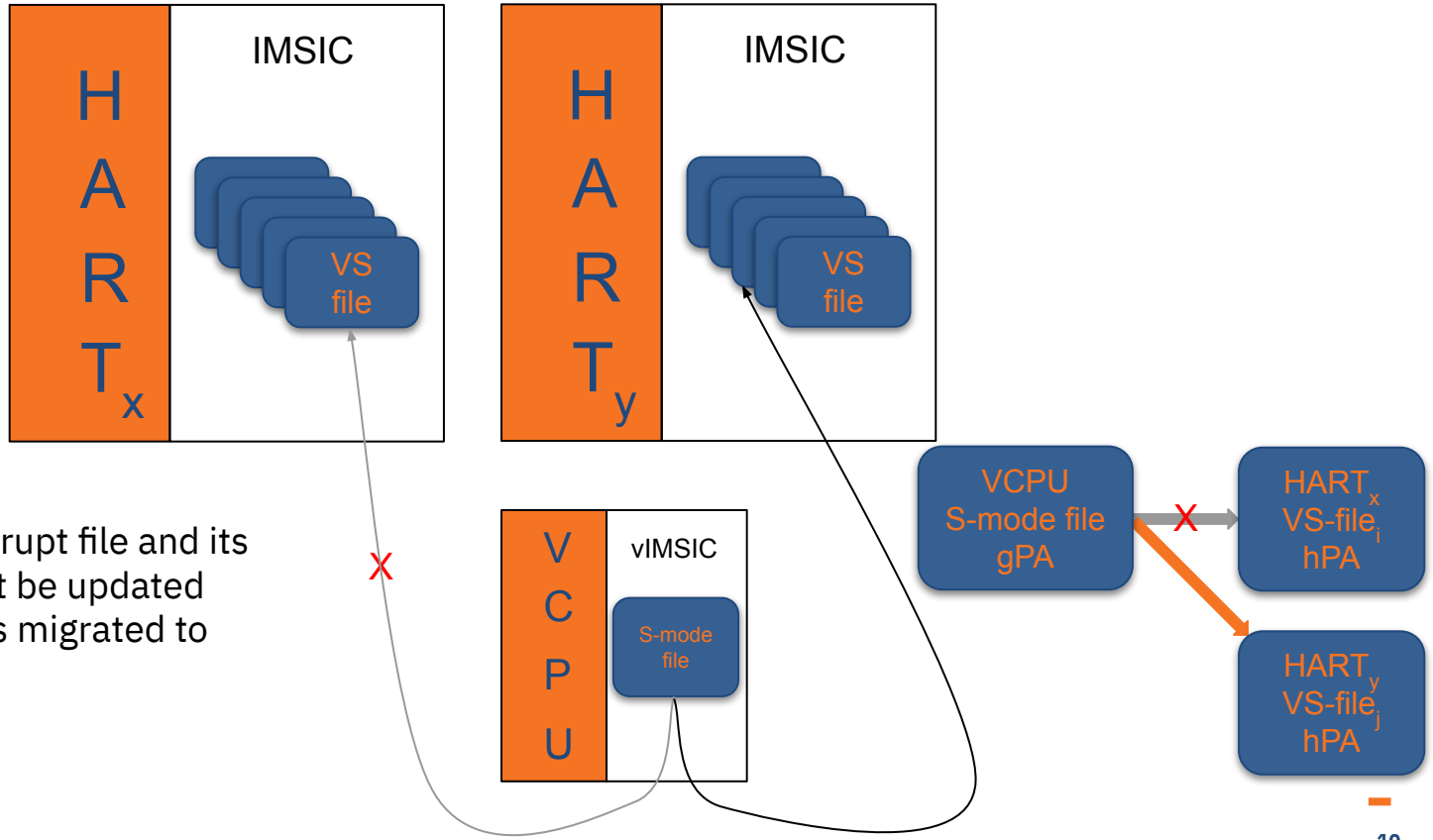
- The maximum number of VS-mode files is 31 for RV32 and 63 for RV64.
  - The server-soc specification currently requires 5.

CPU migration of a VCPU



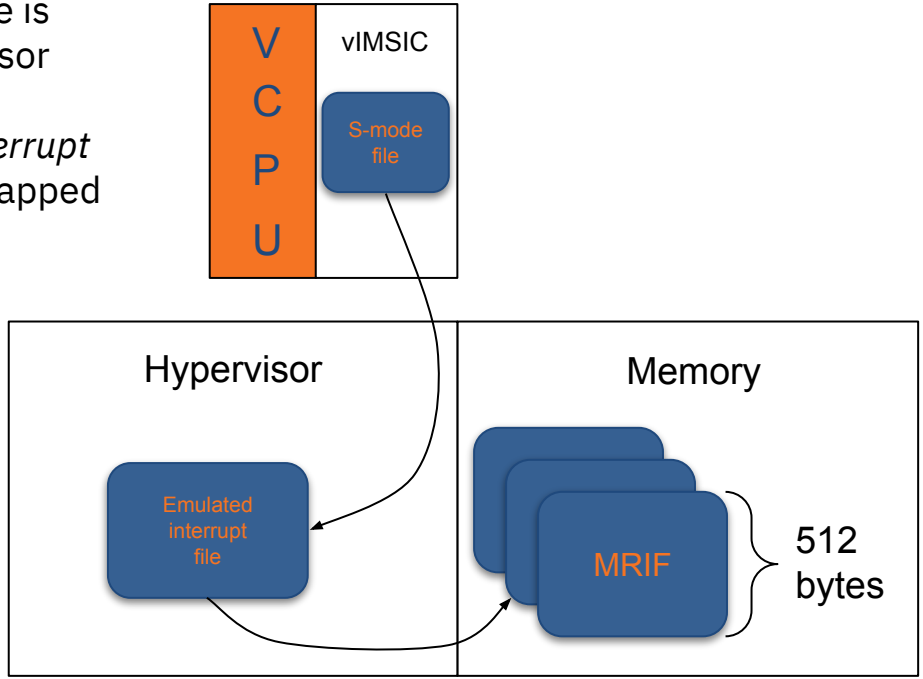
VCPU migration

- The guest interrupt file and its mappings must be updated when a VCPU is migrated to another hart.



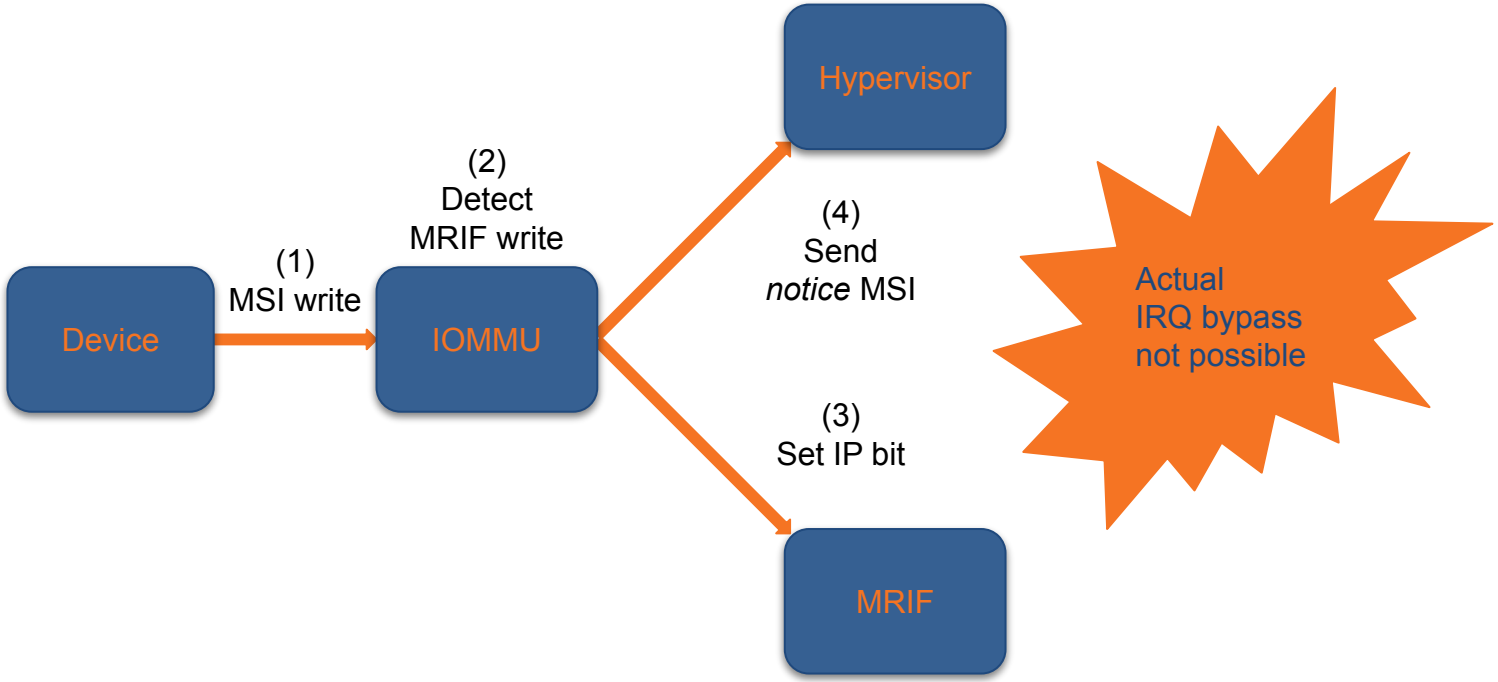
Guest MSI support

- When no VS-mode file is available, the hypervisor provides an MRIF (*memory-resident interrupt file*). Access to it is trapped and emulated.



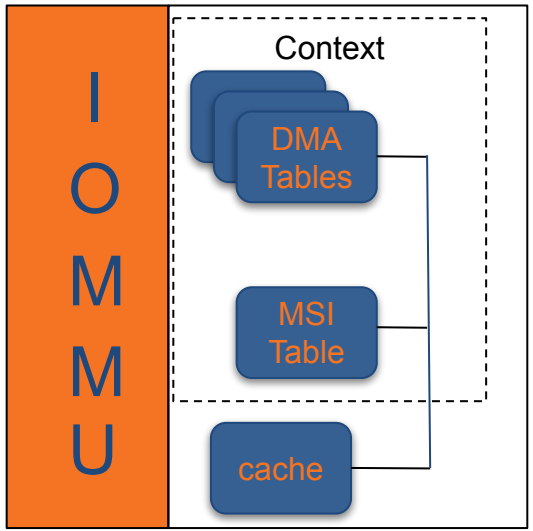
RISC-V IOMMU guest MSI support

VENTANA MICRO
RISC-V Performance Leader



RISC-V IOMMU guest MSI support

VENTANA MICRO
RISC-V Performance Leader



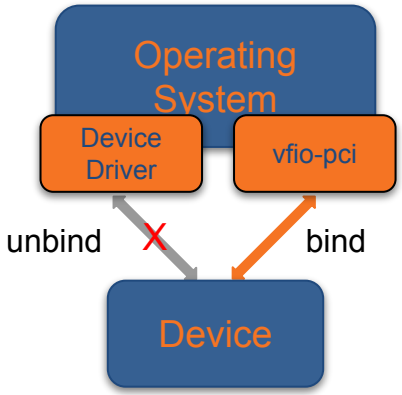
The hypervisor can modify the MSI table independently of subsystems which only modify the DMA tables. (MSI PTEs have a special format.)

Linux and KVM support

VFIO

- VFIO is a Linux framework for assigning devices and
  - provides userspace an ioctl interface to the IOMMU for device assignment
  - is IOMMU / device agnostic and independent of any hypervisor
  - operates on containers of groups of devices
  - device configuration (e.g. guest accesses to PCI config space) still traps to the hypervisor

IOMMUFD also provides userspace an interface and may one day allow VFIO to be deprecated, but it doesn't yet have irqbypass support.



sysfs interface used to bind devices

VFIO's role

- Hypervisor VMM can enable an assigned device to DMA to guest memory

  ```
  ioctl(vfio_container, VFIO_IOMMU_MAP_DMA, &dma_map)
  ```

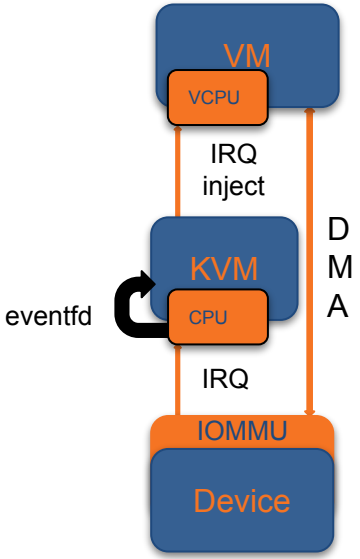
  where *dma_map* describes a memory region.

- And handle IRQ setup on guest configuration traps

  ```
  ioctl(vfio_device, VFIO_DEVICE_SET_IRQS, &irq_set)
  ```

  where *irq_set* describes a range of interrupt IDs and may provide eventfds to trigger upon reception.

  (With eventfds, `irq_bypass_register_producer()` is also called, which we'll come back to later.)



KVM's role

- KVM configures an irqfd to inject an interrupt upon receiving the eventfd signal

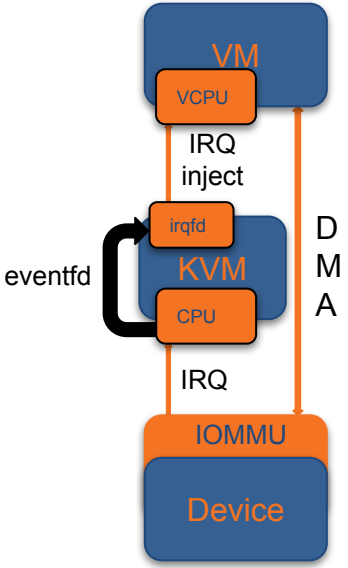
  ```
  ioctl(vm_fd, KVM_IRQFD, &irqfd)
  ```

  where *irqfd* maps an eventfd to an interrupt ID. For an MSI, the interrupt ID (*gsi*) is first configured in an architecture-specific way

  ```
  ioctl(kvm->vm_fd, KVM_SET_GSI_ROUTING, &routing)
  ```

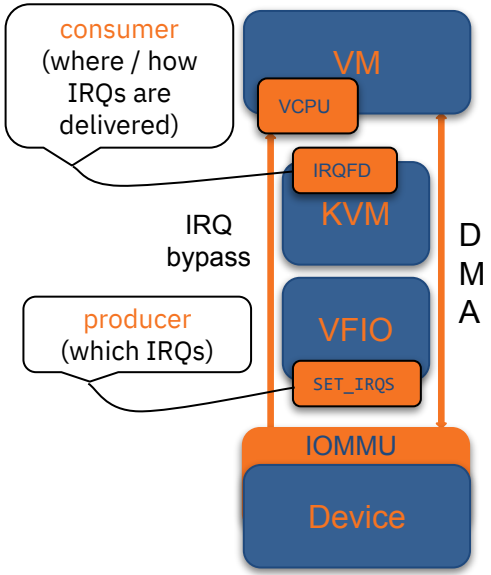
- KVM is still in the IRQ path, but…
- …`KVM_IRQFD` also calls `irq_bypass_register_consumer()`

  (Recall VFIO registered a producer.)



irqbypass

- irqbypass is enabled when both a producer and consumer are registered which have matching tokens (tokens must be unique per pair)
- VFIO unconditionally registers a producer with the eventfd context pointer as its token in `VFIO_DEVICE_SET_IRQS`
- KVM provides an architecture-specific consumer with the same token in `KVM_IRQFD`
- Matching a producer and consumer invokes their callbacks, as does reconfiguration of IRQ routing by subsequent `KVM_SET_GSI_ROUTING` calls

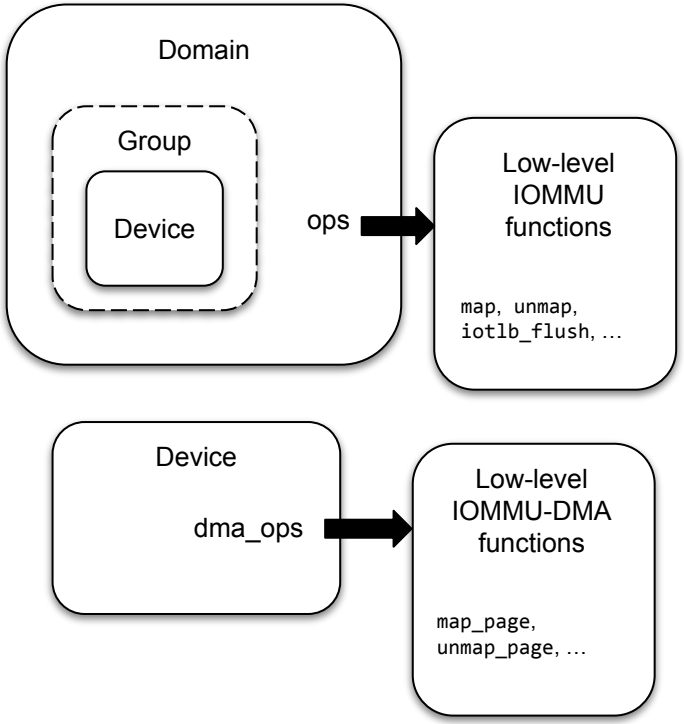


irqbypass

- `KVM_IRQFD`'s irqbypass consumer sets up where / how IRQs are delivered
- For RISC-V
  - **Where** - the interrupt file or MRIF of the VCPU which corresponds to the gPA configured by the guest
  - **How** - instructs the IOMMU to map the gPA to its hPA (also informs the IOMMU as to whether the mapping is an interrupt file or an MRIF)
- "How" brings us to other Linux subsystems

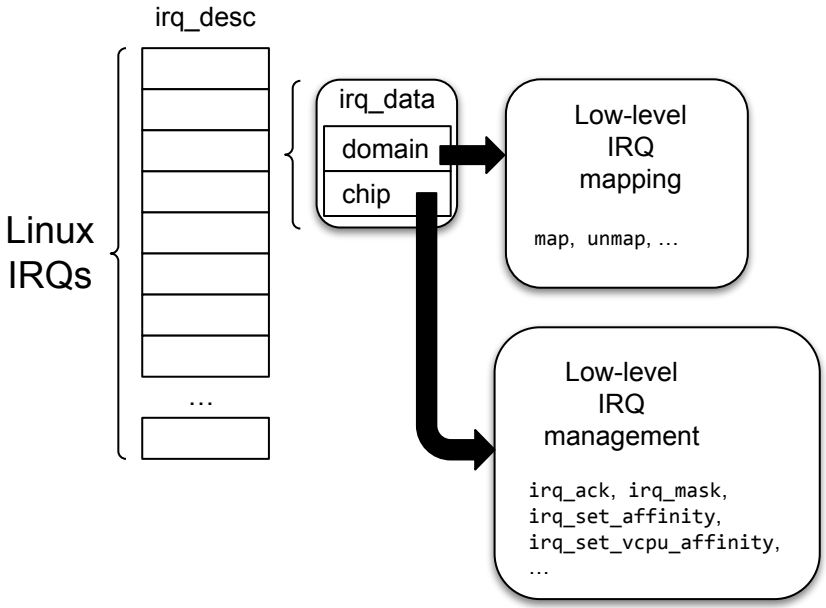
Linux IOMMU subsystem

- Provides IOMMU domain and group abstractions and a generic API to manage domains and devices
- IOMMU domains contain function pointers to IOMMU-specific implementations
- *device* is also an abstraction which is configured to use IOMMU DMA functions when its device is under IOMMU control
- Also provides a couple MSI helpers to map MSI pages and translate MSI messages
  - But MSI page mappings will go through the same paths as the DMA mappings



Linux IRQ subsystem

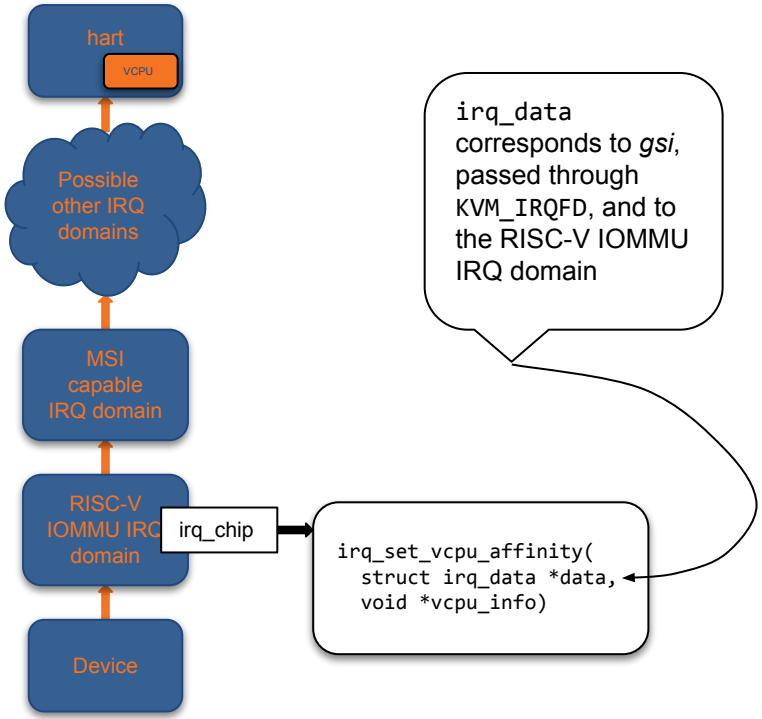
- Linux provides global, abstract Linux IRQ numbers to provide a generic API for managing IRQs
- Interrupt controller-local IRQ numbers (*hwirq*) and hardware description (DT / ACPI) virtual IRQ numbers are mapped to Linux IRQ numbers
- Hierarchical IRQ domains, which match the platform's interrupt controller topology, simplify the Linux IRQ number allocation
- IRQ domains provide functions to map Linux IRQ numbers to local IRQ numbers



An IRQ domain for device assignment

- Upon binding a device to VFIO we need to set up its IRQ domain to one which understands device assignment
  - “understanding device assignment” for IRQs just means the IRQ chip associated with with the IRQ domain needs to implement `irq_set_vcpu_affinity()`
  - For RISC-V, the IOMMU driver should implement `irq_set_vcpu_affinity()`, as it knows how to map guest interrupt files and MRIFs
  - VFIO allocates an IOMMU domain to which it attaches its group of devices
  - IOMMU domains have types. `IOMMU_DOMAIN_UNMANAGED` is defined as “DMA mappings managed by IOMMU-API user, used for VMs” and is the type allocated from VFIO
  - TODO
  - The RISC-V IOMMU driver’s `attach_dev` domain op is the right place to set up the IRQ domain

An IRQ domain for device assignment



- `vcpu_info` defines a protocol between hypervisors and the RISC-V IOMMU driver

<u>An initial proposal</u>

```
struct riscv_iommu_vcpu_info {
        phys_addr_t gpa;
        phys_addr_t hpa;
        bool is_mrif;
};
```

- The KVM irqbypass consumer's `add_producer` callback invokes `irq_set_vcpu_affinity()` on a match with the VFIO producer

TODO

An IRQ domain for device assignment

VENTANA MICRO
RISC-V Performance Leader

- What about CPU migration and guest interrupt file “swapping”?
  - KVM’s `vcpu_load()` runs before entering guest mode
  - KVM tracks which CPU a VCPU last ran on, so it could detect a change during load and issue updates with `irq_set_vcpu_affinity()`
  - Tracking last hPA of the vIMSIC is better though, because it covers CPU migration and KVM guest interrupt file or MRIF target changes
  - When `vcpu_load()` sees a change in the VCPU’s vIMSIC hPA, all IRQs of all assigned devices affined to the VCPU get updated

TODO

TODO

TODO

- RISC-V KVM
  - Patch series 1
    - irqbypass consumer `add_producer` callback which looks up the Linux IRQ associated with *gsi* and invokes `irq_set_vcpu_affinity()` with a new struct (`struct riscv_iommu_vcpu_info`)
    - Check for a VCPU vIMSIC hPA changes in `vcpu_load()` and invoke `irq_set_vcpu_affinity()` on each IRQ which needs an update
  - Patch series 3
    - Modify/extend accounting to differentiate between devices with guest interrupt file irqbypass vs. MRIF irqbypass
- RISC-V IOMMU driver
  - Patch series 2
    - Create an IRQ domain with an IRQ chip which implements `irq_set_vcpu_affinity()`
    - Modify `attach_dev` to set the new IRQ domain for device IRQs when the IOMMU domain type is `IOMMU_DOMAIN_UNMANAGED`

References

- RISC-V IOMMU specification (ratified 1.0)
  https://github.com/riscv-non-isa/riscv-iommu
- RISC-V AIA specification (ratified 1.0)
  https://github.com/riscv/riscv-aia
- RISC-V IOMMU architecture overview
  https://www.youtube.com/watch?v=8fIQqXwGST8
- AIA Virtualization in KVM RISC-V
  https://www.youtube.com/watch?v=r071dL8Z0yo
- Linux RISC-V IOMMU support (under review)
  https://lore.kernel.org/all/cover.1689792825.git.tjeznach@rivosinc.com/
- Linux RISC-V AIA support (under review)
  https://lore.kernel.org/all/20230802150018.327079-1-apatel@ventanamicro.com/
- KVM AIA irqchip (including vIMSIC) support (merged for v6.5)
  https://lore.kernel.org/all/20230615073353.85435-1-apatel@ventanamicro.com/

Thank You