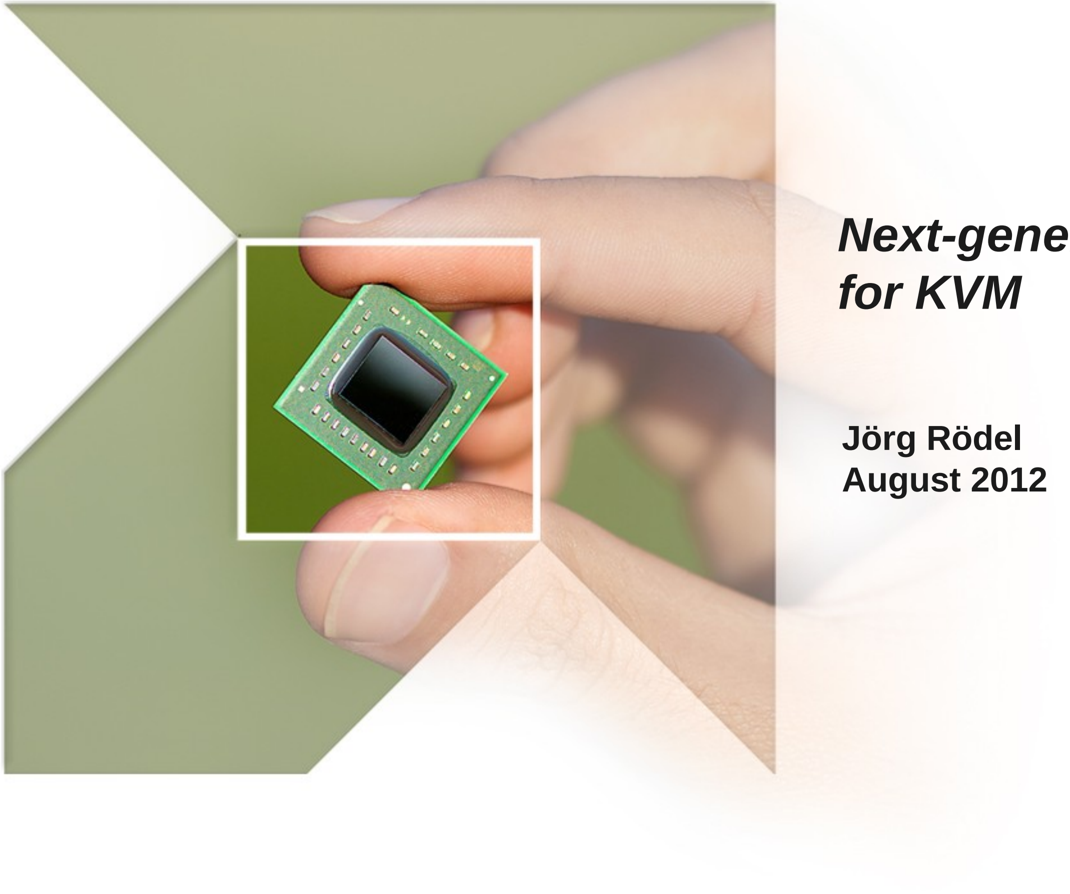


Next-generation Interrupt Virtualization for KVM

Jörg Rödel
August 2012

KVM Interrupt Virtualization Today

- KVM emulates local APIC and IO-APIC
  - All reads and writes intercepted
  - Interrupts can be queued from user space or kernel space
- IPI cost is performance-critical
  - Costs at least two intercepts and a host-level IPI today
  - Without x2apic, even more intercepts
- For device pass-through, interrupt virtualization is performance-critical too
  - Every device interrupt causes an intercept
  - A lot more expensive than bare metal

Hardware Support

- As of today hardware support for interrupt virtualization is very limited
  - On AMD hardware there is direct CR8 (TPR) access for the guest
  - Used only by 64bit guests
- A new hardware feature is planned on AMD:

Advanced Virtual Interrupt Controller

(or AVIC)

Advanced Virtual Interrupt Controller

- AVIC is designed to accelerate the most common interrupt system features for the guest
  - Inter-processor interrupts
  - TPR accesses
  - Interrupts from assigned devices
- AVIC virtualizes the local APIC for each VCPU
  - KVM allocates a virtual APIC backing page (vAPIC page)
  - Guest physical APIC ID table
  - Guest logical APIC ID table
- No support for X2APIC in the initial version

The Doorbell Mechanism

- Doorbell is used to signal AVIC interrupts between physical CPUs
  - Source PCPU figures out physical APIC ID of the destination
  - When destination VCPU is running (IsRunning==1) it sends a Doorbell message
- IOMMU can also send Doorbell messages to PCPUs
  - IOMMU checks if VCPU is running too
  - For not running VCPUs it sends an event log entry
- MSR can also be used to issue Doorbell messages by hand
- When Doorbell is received the pCPU re-evaluates the IRR of the vAPIC page and delivers interrupt as possible and necessary

Guest Virtual APIC Backing Page

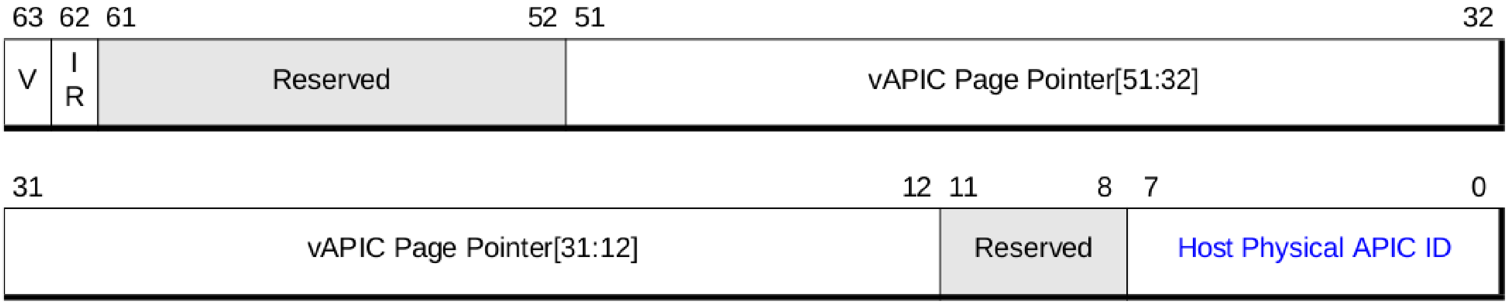
- Used to store local APIC contents for one VCPU
  - Most fields can be read without intercepts
  - Writes to non-accelerated fields cause intercepts
- Currently accelerated fields
  - Offset 0x80: TPR
  - Offset 0xB0: EOI (for edge-triggered interrupts only)
  - Offset 0x300: ICR Low
  - Offset 0x310: ICR High
- Accelerated means that an access is handled in hardware and does not cause an intercept when possible

Running and not-running VCPUs

- When an interrupt needs to be delivered to VCPU, hardware:
  - Looks up the physical CPU (PCPU) the VCPU is running on
  - Sends a Doorbell message to this PCPU
  - The PCPU evaluates IRR of vAPIC page and delivers interrupt to guest if necessary
- If target PCPU is not running software is notified about a new interrupt for this VCPU
  - In case of IPI, with an intercept on the originating VCPU
  - In case of PCI, device interrupt with an IOMMU event log entry

Physical APIC ID Table

- Maps guest physical APIC IDs to host vAPIC pages
- Contains the IsRunning bit which indicates if the VCPU this APIC ID belongs to is currently in guest mode
- If VCPU is running, it also contains the host physical APIC ID of the core it is running on
- This table is entirely maintained by KVM



63	62	61 52	51 32
V	IR	Reserved	vAPIC Page Pointer[51:32]

31 12	11 8	7 0
vAPIC Page Pointer[31:12]	Reserved	Host Physical APIC ID

Logical APIC ID Table

- Maps guest logical APIC IDs to guest physical APIC IDs
  - Indexed by guest logical APIC ID
- Supports different address modes
  - Flat mode
  - Cluster mode
- Maintained by KVM, too
- Entry is simpler than in the physical APIC ID table

31	30 8	7 0
V	Reserved	Guest Physical APIC ID

Support in the IOMMU

- For AVIC accelerated device pass-through, the IOMMU is necessary
  - IOMMU delivers interrupt directly to destination PCPU using Doorbell message
- The interrupt remapping table set-up is changed for assigned devices
  - By design, the MSI(X) capabilities are still managed by KVM
- A separate IsRunning bit is maintained for the IOMMU
  - Performance reasons
  - Minimize the system memory data structures the IOMMU needs to access

AVIC Support in KVM

- Support can be implemented mostly in the KVM-AMD module
- Some changes to the current local APIC emulation necessary
  - Change the layout of the APIC register data structure to match its offsets with the real local APIC register offsets
  - KVM x86 core code will allocate the vAPIC pages
- vAPIC page needs to be mapped in the nested page table
  - Problematic
  - Likely requires some changes in the KVM SoftMMU code
- Set-up of VMCB, physical, and logical APIC ID tables happens in the KVM module

AVIC Support in KVM for Device Pass-through

- Changes to VFIO required
  - When destination VCPU is running direct delivery should be configured
  - For not-running VCPUs, the current mechanism should still work
- Ideally, this is fully transparent to user space
- Details are not worked out yet

Questions?

Trademark Attribution