Enabling Optimized Interrupt/APIC Virtualization in KVM

Jun Nakajima

Intel Open Source Technology Center

November 8, 2012

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL® PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.

Intel may make changes to specifications and product descriptions at any time, without notice.

All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Agenda

- **Overview of Hardware Enhancements**
- **Optimized Interrupt/APIC Virtualization**
  - APIC-Register Virtualization
  - Virtual-Interrupt Delivery
  - Posted-Interrupt Processing
- **Enabling for KVM**
- **Summary**

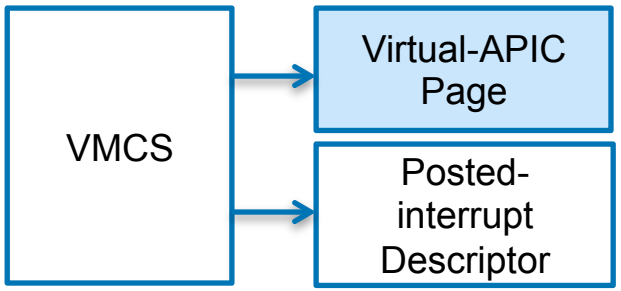
Interrupt/APIC Virtualization: Overview

- **VMM must virtualize guest's interrupts and interrupt controller (APIC)**
  - Models APIC control state on a "virtual-APIC page" in memory
- **VMM must emulate nearly all guest accesses to APIC control registers**
  - Require VM exits and entries
  - Decode and emulate guest instructions that access APIC (except x2APIC mode)
  - Except for Intel® VT FlexPriority, which virtualizes access to one APIC control register
    - Task priority – TPR
    - No VM exits required in this case
- **VMM must virtualize all interrupts coming to guest**
  - Must determine when guest is ready to receive interrupts and deliver as needed
- **Virtualization of interrupts and APIC is a major source of overhead**

Motivations for Further Optimizations

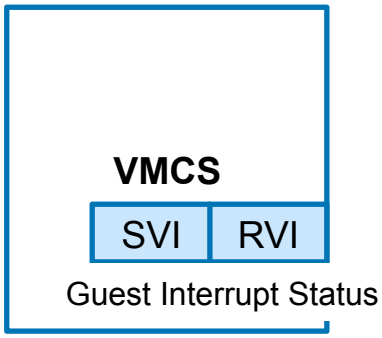
- **Reduce unique overheads of virtualization**
  - Intel is fanatically committed
- **Virtualization has come to be default deployment platform for IT**
  - Any application, even most performance demanding, may run in virtualization
- **Virtualization is foundation of Cloud**
  - More I/O performance/scalability for Web apps, Database, Big Data, HPC, etc.
- **Optimized Interrupt/APIC Virtualization is helpful for nested virtualization**
  - VMMs typically have more timer and I/O interrupts.

APIC-Register Virtualization



- **Virtualized APIC-Read Accesses:**
  - Emulate APIC-access, **w/o causing VM exit**
    - Read data from Virtual-APIC page at page offset specified APIC-access page
    - 02H (local APIC ID), 03H (local APIC version), 08H (task priority), …,
- **Virtualized APIC-Write Accesses:**
  - Write data to Virtual-APIC page at page offset specified APIC-access page
  - Emulate APIC-access, **w/o causing VM exit**
    - 08H (task priority), 0BH (end of interrupt), …, 30H and 31H (interrupt command),…
  - APIC-write VM Exit (new VM exit)
    - Transform **fault-like** APIC-access VM exits into **trap-like** APIC-write VM exits
- **Include virtual x2APIC mode support**

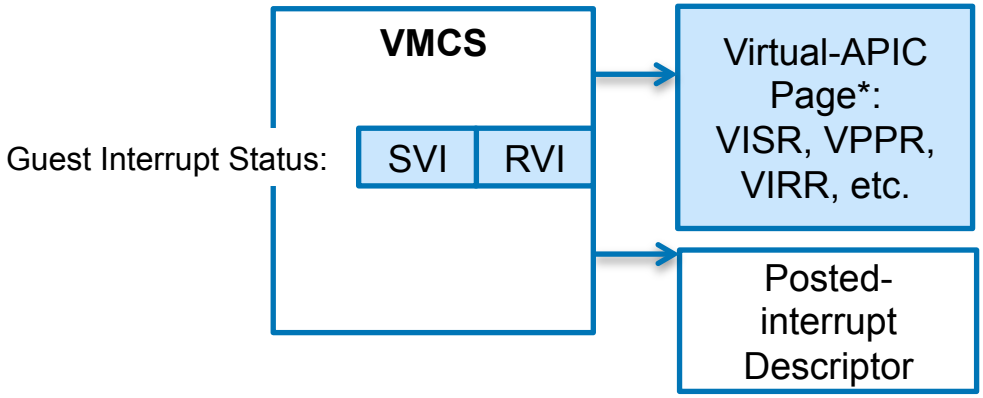
Virtual-Interrupt Delivery



- **Evaluation of pending virtual interrupts:**
  - VM Entry, TPR virtualization, EOI virtualization, self-IPI virtualization, and posted-interrupt processing
- **Once recognized, a virtual interrupt may be delivered in guest:**
  - "interrupt-window exiting" VM-execution control needs to 0
  - Eliminate "interrupt-window exiting"
- **Deliver virtual interrupts w/o VM exit:**
  - Updates **guest interrupt status** (new 16-bit VMCS field) and delivers event within guest
    - Requesting virtual interrupt (RVI) (low byte) – vector of the highest priority virtual interrupt that is **requesting service**
    - Servicing virtual interrupt (SVI) (high byte) – vector of the highest priority virtual interrupt that is **in service**

Virtual-Interrupt Delivery (Details)

Vector ← RVI;
VISR[Vector] ← 1;
SVI ← Vector;
VPPR ← Vector & F0H;
VIRR[Vector] ← 0;
If (any bits set in VIRR)
    RVI ← highest index of bit set in VIRR;
else
    RVI ← 0;
Deliver interrupt with Vector through IDT;
Cease recognition of any pending virtual interrupt;



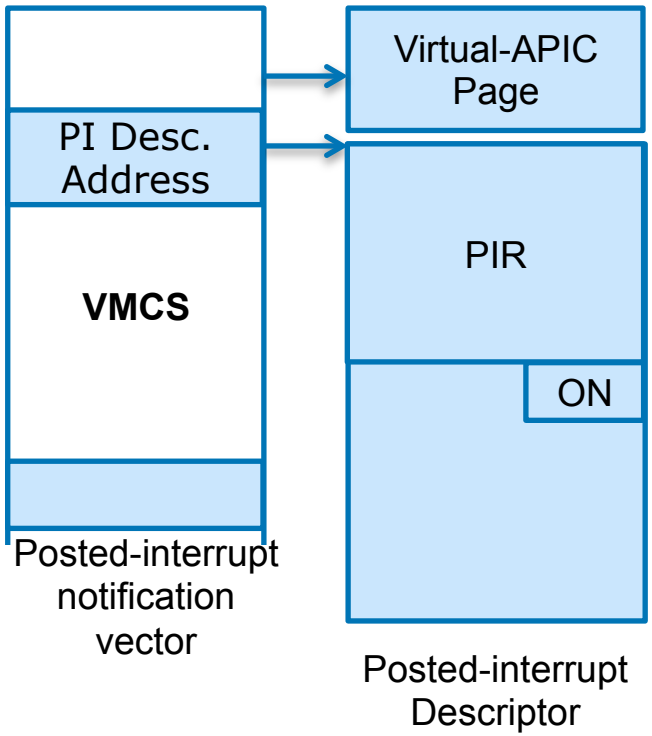
- *: VISR (Virtual interrupt-service register), VPPR (Virtual processor-priority register), VIRR (Virtual interrupt-request register) are in **Virtual-APIC page**

EOI Virtualization

- **Key to reduce virtualization overhead**
  - Every interrupt causes access to APIC EOI register (offset 0x0B0)
- **EOI-Exit Bitmap (in VMCS) determines which virtualized writes cause VM exits:**
  - EOI_EXIT*n* (n=0, 1, 2, and 3) vectors from *n*64* to *(n+1)*64 – 1*
  - Allows to avoid VM exits (if applicable) if the bit is not set
  - VM exits can be avoided for many cases
- **EOI virtualization uses and updates Guest Interrupt Status (SVI)**
- **Evaluate pending virtual interrupts**
  - Virtual interrupts are delivered accordingly if recognized
- **Support x2APIC mode**

Posted-Interrupt Processing

- **Sending notification (IPI) w/o VM exit**
  - If physical vector == Posted-interrupt notification vector (VMCS field)
- **Process the virtual interrupts by recording them as pending on Virtual-APIC page**
- **Record virtual interrupts in Posted-Interrupt Descriptor**
  - Clears ON*
  - Perform logical-OR of PIR* into VIRR and clears PIR
  - RVI to be maximum of old value of RVI and highest index of all bits that were set in PIR
- **Software needs to set PIR bits for guest vectors in advance**
- ***ON: Outstanding Notification, PIR: Posted-Interrupt Request (format on next page)**



Posted-interrupt Descriptor

- **In general, data structures referenced by VMCS shouldn't be modified when guest is running**
- **The general requirement doesn't apply to Posted-interrupt Descriptor field**
  - Use locked read-modify-write instructions to modify

Bit Position(s)	Name	Description
255:0	Posted-interrupt requests	One bit for each interrupt vector. There is a posted-interrupt request for a vector if the corresponding bit is 1
256	Outstanding notification	If this bit is set, there is a notification outstanding for one or more posted interrupts in bits 255:0
511:257	Reserved for software and other agents	These bits may be used by software and by other agents in the system (e.g., chipset). The processor does not modify these bits.

Enabling APIC-Register Virtualization in KVM

- **What KVM does in software today:**
  - Handled by lapic.c
  - Maintains virtual APIC state in "APIC Register Page" (defined in kvm_lapic structure), i.e. Virtual - APIC page. The register layout is same as actual local APIC
    - `void *regs;`

- **Changes for APIC-Register Virtualization:**
  - Processor simply accesses virtual APIC page w/o VM exit if APIC-Register virtualization is enabled
    - Set 1 in "APIC-register virtualization" VM-execution control
  - Use trap-like APIC-write VM exits to avoid "decode & emulate"
    - New VM Exit Reason: **APIC write (56)**. "Guest software completed a write to the Virtual-APIC page that must be virtualized by VMM software"

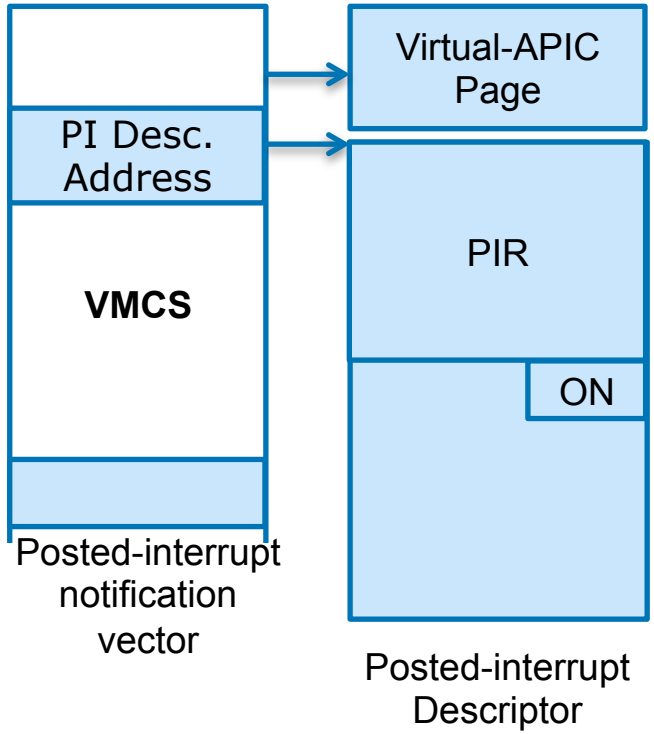
Enabling Virtual-Interrupt Delivery in KVM

- **What KVM does in software today:**
  - Emulates APIC with the state maintained in Virtual-APIC page
  - Evaluates pending virtual interrupts
    - Manually injects virtual interrupts at VM entry time
  - If interrupts are disabled by the guest, the processor comes back when ready
    - Use Interrupt-window VM exit

- **Changes required for Virtual-Interrupt Delivery:**
  - Have the processor modify Virtual-APIC page to maintain virtual APIC state
  - Have the processor evaluate pending virtual interrupts and deliver
    - VM entry causes evaluation of pending virtual interrupts
  - If interrupts are disabled by the guest, the processor delivers virtual interrupt without VM exit
    - Don't use Interrupt-window VM exit
  - EOI Virtualization
    - Set up EOI-Exit Bitmap

Usage Examples and Enabling Posted-Interrupt Processing in KVM

- **Injecting interrupts to VCPU(s) from remote CPU without causing VM exit on the destination**
  - Guest IPI
  - Virtual device interrupts from remote CPU (incl. to frontend from backend)
- **Enabling Guest IPI:**
  - Allocate a vector for event notification on the host
  - Write the vector to “Posted-interrupt notification vector” VMCS field
  - Set a bit in PIR of the destination, corresponding to the guest vector for IPI upon VM exit caused by access to ICR (Interrupt Command Register) in the guest
  - Send the even notification



Summary

- **Optimized Interrupt/APIC Virtualization**
  - APIC-Register Virtualization, Virtual-Interrupt Delivery, and Posted-Interrupt Processing
- **Enabling the features for KVM**
- **Current status**
  - Patches have been submitted, and got helpful comments from Avi, etc.:
    - APIC-Register Virtualization
    - Virtual-Interrupt Delivery
  - Working on optimized “Guest IPI” using Posted-Interrupt Processing
    - Got it working; no VM exits on the destination
- **Preliminary results (Netperf)**
  - Eliminate up to 50% of VM exits (most of those related to virtualization of interrupts/APIC)
  - Optimize up to 10% of VM exits (emulation made easier for some APIC writes)