KVM SMM

Xiao Guangrong
<guangrong.xiao@linux.intel.com>

Index

- SMM Introduction
- SMM Virtualization
- SMM in ACPI
- SMM in UEFI

SMM Introduction

- SMM: System management mode
  - Enter by SMI (System management interrupt)
  - Exit by RSM instruction
- SMRAM, the memory region, is only visible in SMM
  - 64K size, begin with 0x30000 on default, can be relocated by setting SMBASE
  - Is used to save/load context when do mode-switch, save SMM info (e.g. SMI handler) and other OEM info

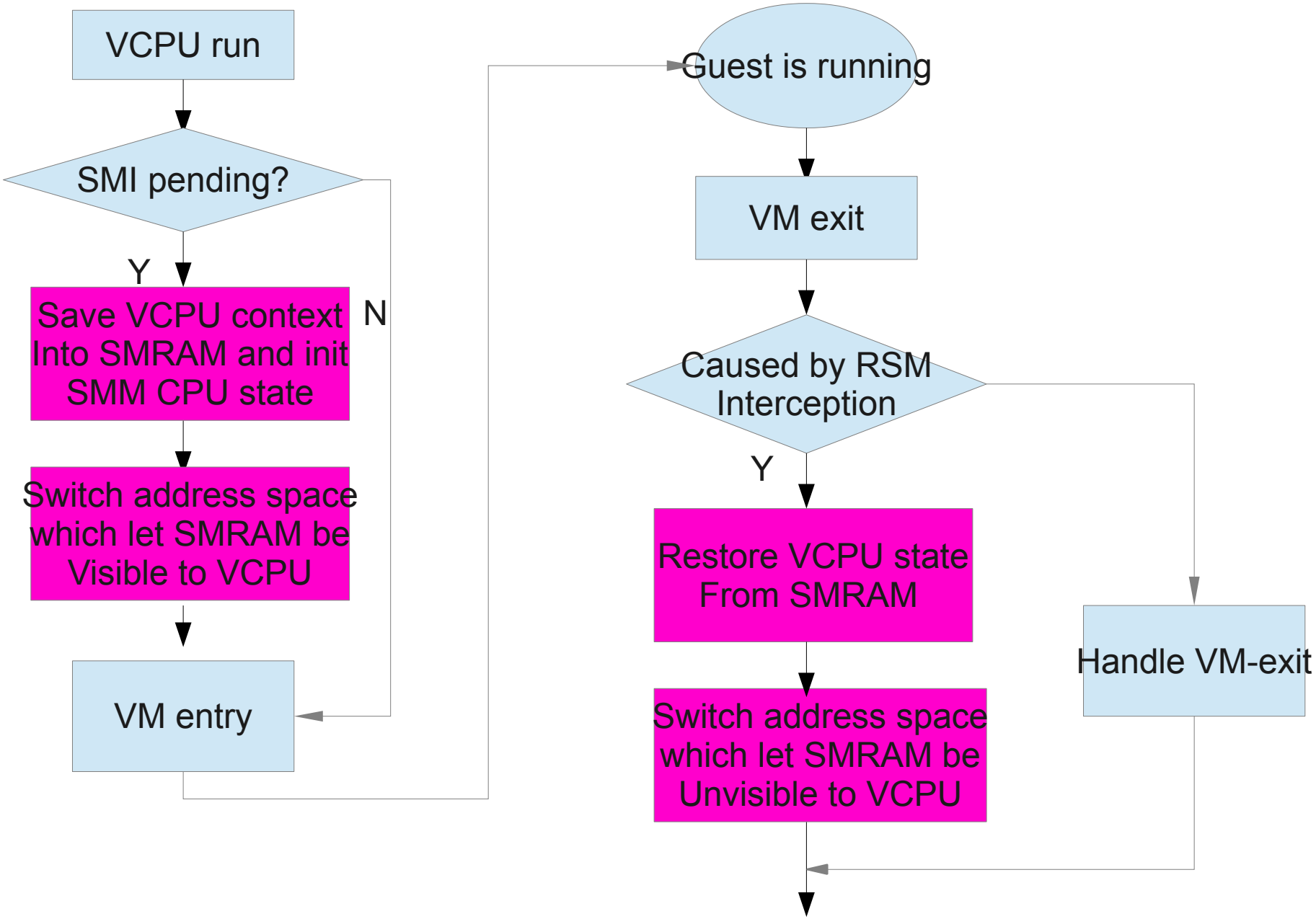
SMM Introduction (cont.)

- Unified state when entering SMM
  - Real mode
  - Addressable range 0 ~ 4G
  - segment limit is 4G
  - CS:EIP points to SMBASE + 8000H
  - EFLAGS.IF = EFLAGS.TF = 0 and NMI, SMI, A20 interrupts are blocked
  - IDT is not initialized
- More info please refer to SDM Vol 3. 34

SMM implementation

- Overview
- New API introduced
- Entering SMM Virtualization
- Exiting SMM Virtualization
- SMRAM Virtualization

SMM implementation: Overview



SMM Implementation

- New API introduced
  - KVM_SMI
    - Queues an SMI on the VCPU
  - Is used to
    - Inject SMI if vAPIC is in userspace
    - Emulate ICH9/Piix4 SMI port (0xb2) access (Write appropriate value to this port will cause SMI).

SMM Implementation (cont.)

- Entering SMM Virtualization
  - If KVM_SMI is issued or LAPIC accepts a SMI interrupt, it queues a SMI request on the VCPU
  - When VCPU gets scheduled, it stores current VCPU context into SMRAM and initializes VCPU to the SMM init state, these completely follow SDM
  - Switch MMU to let SMRAM be visible (more detail in follow pages)
  - Enter guest

SMM Implementation (cont.)

- Exiting SMM Virtualization
  - If RSM instruction is intercepted, it restores VCPU context from SMRAM
  - Switch MMU to let SMRAM not be visible (more detail in follow pages)
  - Enter guest

SMM Implementation: SMRAM Virtualization

- Multiple address spaces
  - Extend KVM_SET_MEMORY_REGION, KVM_GET_DIRTY_LOG … APIs:
    - bits 16-31 of "slot" specifies the address space which is being modified
    - Userspace is able to handle multiple address spaces by using these APIs
  - Private memslots are always registered for all address spaces, such as APIC access page
- Two address spaces are used
    - Space 0 for normal mode (!smm)
    - Space 1 for SMM mode, Space 1 = Space 0 + SMRAM

SMM Implementation: SMRAM Virtualization (cont.)

- VCPU chooses address space based on its mode, if it's on normal mode use space 0, use 1 if it is on SMM mode
- Shadow page tables are separated, switch to the SMM page table when VCPU enters SMM mode, switch it back when VCPU leaves SMM mode

SMM in ACPI

- SMM related functions are defined in FADT (Fixed ACPI Description Table)
  - SMI_CMD
    - System port address of the SMI Command Port
  - Context command, the context written to SMI_CMD
    - ACPI_ENABLE
    - ACPI_DISABLE
    - S4BIOS_REQ
    - PSTATE_CNT

SMM in ACPI (cont.)

- Detailed SMM events
  - ACPI_ENABLE / ACPI_DISABLE, enable / disable SMM mode
    - e.g, on the platform which has PIIX4:
    - SMI_CMD = 0xb2
    - ACPI_ENABLE = 0xf1
    - write 0xf1 to ioport 0xb2 to enable SMM
  - S4BIOS_REQ, enter the S4 state
  - PSTATE_CNT, assume Pstate control responsibility
- Currently, S4BIOS_REQ and PSTATE_CNT are never used in Qemu
- ACPI_ENABLE / ACPI_DISABLE is inhibited for kvm_enable, Paolo turned it on in his Qemu part patchset

SMM in UEFI

- SMM defined in SMM Communication ACPI Table
  - This table describes a special software SMI that can be used to initiate inter-mode communication in the OS present environment by non-firmware agents with SMM code

SW SMI Number	4	54	Number to write into software SMI triggering port.
Buffer Ptr Address	8	58	Address of the communication buffer pointer. The pointer address (this field) and the pointer value (the actual address of the communication buffer) are 64-bit physical addresses. The creator of this table must initialize pointer value with 0. The communication buffer must be prefixed with the **EFI_SMM_COMMUNICATE_HEADER** defined in the "Related Definitions" section below.

SMM in UEFI (cont.)

  - SW SMI Number: the ioport used to trigger SMI
  - Buffer Ptr Address: the communication buffer
- Internal SMM communication and SMM Driver ↔ Runtime/Boot servers are defined in PI SMM SPEC
- How does UEFI use this SMM mechanism?
  - NOT DEFINED
  - Let find the answer in the source code by ourselves. :)
  - Follow usages are based on Intel EDK2/OVMF

SMM in UEFI: SMM in Intel EDK2/OVMF

- The foundation of Variable Services
  - Variables are isolated and protected by SMM, read / write variable operations communicate with SMM
- The foundation of Firmware Volume Block (FVB)
  - Image is isolated by FVB, read / write access image block operations communicate with SMM
- The foundation of EDK2 DP Reporting Utility and Firmware Performance Data Table (FPDT)
  - TimeStamp is tracked in SMM and performance statistics is reported by SMM
  - SMRAM profiling

SMM in UEFI: SMM in Intel EDK2/OVMF (cont.)

- The foundation of Fault Tolerant Write (FTW)
  - FTW is used to avoid the situation that during variable reclaim, the flash block will be erased and be written again. if power lost or system shutdown, the variable firmware volume will be partially destroyed.
  - FTW omits the risk by writing data to spare block, then context can be restored from the spare block if any exception happened.
  - The spare block and other info are under the control of SMM

Thanks! :0