In early 2000 it became obvious that hardware support for virtualization was necessary, and Intel and

AMD started work on the first-generation virtualization extensions of the x86 3 architecture. In 2005

Intel released two Pentium 4 models supporting VT-x, and in 2006 AMD announced Pacifica and then

several Athlon 64 models.

A 2006 paper [253] analyzes the challenges to virtualizing Intel architectures and then presents

VT-x and VT-i virtualization architectures for x86 and Itanium architectures, respectively. Software

solutions at that time addressed some of the challenges, but hardware solutions could improve not only

performance but also security and, at the same time, simplify the software systems. We first examine

the problems faced by virtualization of the x86 architecture:

• Ring deprivileging. This means that a VMM forces the guest software, the operating system, and

the applications to run at a privilege level greater than 0. Recall that the x86 architecture provides

four protection rings at levels 0–3. Two solutions are then possible: (a) The (0/1/3) mode, in which

the VMM, the OS, and the application run at privilege levels 0, 1, and 3, respectively; or (b) the

(0,3,3) mode, in which the VMM, a guest OS, and applications run at privilege levels 0, 3, and 3,

respectively. The first mode is not feasible for x86 processors in 64-bit mode, as we shall see shortly.

• Ring aliasing. Problems created when a guest OS is forced to run at a privilege level other than that

it was originally designed for. For example, when the CR register4 is PUSHed, the current privilege

level is also stored on the stack [253].

• Address space compression. A VMM uses parts of the guest address space to store several system

data structures, such as the interrupt-descriptor table and the global-descriptor table. Such data

structures must be protected, but the guest software must have access to them.

• Nonfaulting access to privileged state. Several instructions, LGDT, SIDT, SLDT, and LTR that

load the registers GDTR, IDTR, LDTR, and TR, can only be executed by software running at

privilege level 0, because these instructions point to data structures that control the CPU operation.

Nevertheless, instructions that store from these registers fail silently when executed at a privilege

level other than 0. This implies that a guest OS executing one of these instructions does not realize

that the instruction has failed.

• Guest system calls. Two instructions, SYSENTER and SYSEXIT, support low-latency system calls.

The first causes a transition to privilege level 0, whereas the second causes a transition from privilege

level 0 and fails if executed at a level higher than 0. The VMM must then emulate every guest

execution of either of these instructions, which has a negative impact on performance.

• Interrupt virtualization. In response to a physical interrupt, the VMM generates a “virtual interrupt”

and delivers it later to the target guestOS. But every OS has the ability to mask interrupts5; thus the virtual

interrupt could only be delivered to the guestOS when the interrupt is not masked. Keeping track

of all guestOS attempts to mask interrupts greatly complicates the VMM and increases the overhead.

• Access to hidden state. Elements of the system state (e.g., descriptor caches for segment registers)

are hidden; there is no mechanism for saving and restoring the hidden components when there is a

context switch from one VM to another.

• Ring compression. Paging and segmentation are the two mechanisms to protect VMM code from

being overwritten by a guest OS and applications. Systems running in 64-bit mode can only use

paging, but paging does not distinguish among privilege levels 0, 1, and 2, so the guest OS must run

at privilege level 3, the so-called (0/3/3) mode. Privilege levels 1 and 2 cannot be used; thus the

name ring compression.

• Frequent access to privileged resources increases VMM overhead. The task-priority register (TPR)

is frequently used by a guest OS. The VMM must protect the access to this register and trap all

attempts to access it. This can cause a significant performance degradation.

Similar problems exist for the Itanium architecture discussed in Section 5.10.

A major architectural enhancement provided by the VT-x is the support for two modes of operations

and a new data structure called the virtual machine control structure (VMCS), including host-state and

guest-state areas (see Figure 5.5):

• VMX root. Intended for VMM operations and very close to the x86 without VT-x.

• VMX nonroot. Intended to support a VM.

When executing a VM entry operation, the processor state is loaded from the guest-state of the VM

scheduled to run; then the control is transferred from the VMM to the VM. A VM exit saves the processor

state in the guest-state area of the running VM; then it loads the processor state from the host-state area

and finally transfers control to the VMM. Note that all VM exit operations use a common entry point

to the VMM.

Each VM exit operation saves the reason for the exit and, eventually, some qualifications in VMCS.

Some of this information is stored as bitmaps. For example, the exception bitmap specifies which one

of 32 possible exceptions caused the exit. The I/O bitmap contains one entry for each port in a 16-bit

I/O space.

The VMCS area is referenced with a physical address and its layout is not fixed by the architecture

but can be optimized by a particular implementation. The VMCS includes control bits that facilitate

the implementation of virtual interrupts. For example, external-interrupt exiting, when set, causes the

execution of a VM exit operation; moreover, the guest is not allowed to mask these interrupts. When the

interrupt window exiting is set, a VM exit operation is triggered if the guest is ready to receive interrupts.

Processors based on two new virtualization architectures, VT-d 6 and VT-c, have been developed.

The first supports the I/O memory management unit (I/O MMU) virtualization and the second supports

network virtualization.

Also known as PCI pass-through, I/O MMU virtualization gives VMs direct access to peripheral

devices. VT-d supports:

• DMA address remapping, which is address translation for device DMA transfers.

• Interrupt remapping, which is isolation of device interrupts and VM routing.

• I/O device assignment, in which an administrator can assign the devices to a VM in any configuration.

• Reliability features, which report and record DMA and interrupt errors that may otherwise corrupt

memory and impact VM isolation.

Next we discuss Xen, a widely used VMM or hypervisor.