Three classes of fundamental abstractions – interpreters, memory, and communications links – are necessary

to describe the operation of a computing system [312]. The physical realization of each one of

these abstractions, such as processors that transform information, primary and secondary memory for

storing information, and communication channels that allow different systems to communicate with one

another, can vary in terms of bandwidth,1 latency,2 reliability, and other physical characteristics. Software

systems such as operating systems are responsible for the management of the system resources –

the physical implementations of the three abstractions.

Resource management, discussed in depth in Chapter 6, grows increasingly complex as the scale

of a system as well as the number of users and the diversity of applications using the system increase.

Resource management for a community of users with a wide range of applications running under

different operating systems is a very difficult problem. Resource management becomes even more

complex when resources are oversubscribed and users are uncooperative. In addition to external factors,

resource management is affected by internal factors, such as the heterogeneity of the hardware and

software systems, the ability to approximate the global state of the system and to redistribute the load,

the failure rates of different components, and many other factors.

The traditional solution for a data center is to install standard operating systems on individual

systems and rely on conventional OS techniques to ensure resource sharing, application protection,

and performance isolation. System administration, accounting, security, and resource management are

very challenging for the providers of service in this setup; application development and performance

optimization are equally challenging for the users.

The alternative is resource virtualization, a technique analyzed in this chapter. Virtualization is a

basic tenet of cloud computing – that simplifies some of the resource management tasks. For example,

the state of a virtual machine (VM) running under a virtual machine monitor (VMM) can be saved

and migrated to another server to balance the load. At the same time, virtualization allows users to

operate in environments with which they are familiar rather than forcing them to work in idiosyncratic

environments.

Resource sharing in a virtual machine environment requires not only ample hardware support and, in

particular, powerful processors but also architectural support for multilevel control. Indeed, resources such as CPU cycles, memory, secondary storage, and I/O and communication bandwidth are shared

among several virtual machines; for each VM, resources must be shared among multiple instances of

an application.

We start our discussion with a look at virtualization principles and the motivation for virtualization.

Then we discuss the interfaces that define the properties of the system at different levels

of abstraction: the application programming interface (API), the application binary interface (ABI),

and instruction set architecture (ISA). We discuss alternatives for the implementation of virtualization

in Sections 5.3 and 5.4, then analyze their impact on performance and security isolation in

Section 5.5.

Two distinct approaches for virtualization, the full virtualization and the paravirtualization, are discussed

in Section 5.6. Full virtualization is feasible when the hardware abstraction provided by the

VMM is an exact replica of the physical hardware. In this case any operating system running on the

hardware will run without modifications under the VMM. In contrast, paravirtualization requires some

modifications of the guest operating systems because the hardware abstraction provided by the VMM

does not support all the functions the hardware does.

Traditional processor architectures were conceived for one level of control because they support two

execution modes, the kernel and the user mode. In a virtualized environment all resources are under the

control of a VMM and a second level of control is exercised by the guest operating system. Although

two-level scheduling for sharing CPU cycles can be easily implemented, sharing of resources such as

cache, memory, and I/O bandwidth is more intricate. In 2005 and 2006 the x86 processor architecture

was extended to provide hardware support for virtualization, as discussed in Section 5.7.

We analyze the Xen VMM in Section 5.8 and discuss an optimization of its network performance

in Section 5.9. High-performance processors (e.g., Itanium) have multiple functional units but do not

provide explicit support for virtualization, as shown in Section 5.10.

The system functions critical for the performance of a VM environment are cache and memory

management, handling of privileged instructions, and input/output (I/O) handling. Important sources for

the performance degradation in a VMenvironment are the cache misses, as we shall see in Section 5.11.

We analyze the security advantages of virtualization in Section 9.6 and some of the potential risks in

Section 5.12. Finally, we discuss software fault isolation in Section 5.13.