Virtualization

By rapidly switching between threads and processes, the illusion of parallelism is created. This seperation between single CPU and being able to pretend there are more can be extended to other resources as well, leading to what is known as **resource virtualization**.

Principle of Virtualization

There are many different types of interfaces, ranging from basic instruction set as offered by a CPU to the vast collection application programming interfaces that are shipped with many current middleware systems.

In its essence, virtualization deals with extending or replacing an existing interface so as to mimic the behaviour of another system. (an interface layer on top of another, for example, WSL on windows with A = Linux, B = Windows)

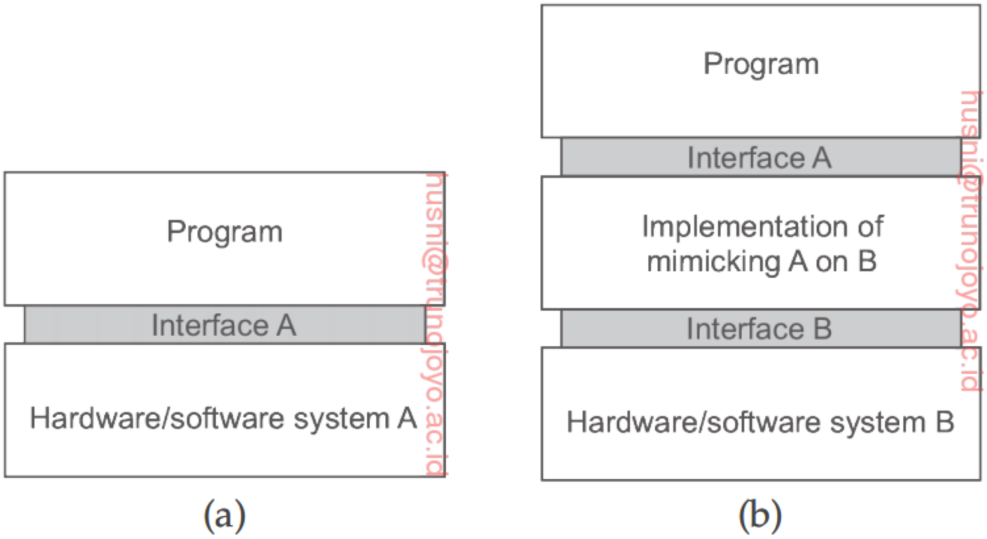
Program
Interface A
Hardware/software system A
(a)
Program
Interface A
Implementation of mimicking A on B
Interface B
Hardware/software system B
(b)

Figure 3.6: (a) General organization between a program, interface, and system. (b) General organization of virtualizing system A on top of B.

Virtualization and distributed systems

Running legacy software on expensive hardware. We are facing a situation in which legacy software cannot be maintained in the same pace as the platform it relies on. Virtualization can help there by porting the legacy software on the new platforms and thus immediately open the latter for large classes of existing programs.

Also, networking has become pervasive. In practice, this connectivity requires that system administrators maintain a large and heterogeneous collection of server computers, each running very different applications, which can then be accessed by clients. In this case, virtualization can help a lot: the diversity of platforms and machines can be reduced by essentially letting each application run on its own virtual machine, possibly including the related libraries and operating systems, which, in turn, run on a common platform.

In last type of virtualization, edge servers in CDNs can support virtualization, allowing a complete site, including its environment to be completely copied. It provides a high degree of portability and flexiblity.

Types of Virtualization

Types of Interface

1. An interface between the software and the hardware, called the **Instruction Set Architecture (ISA)**, forming the set of machine instructions. Divided into 2 subsets:
   - Priveledged instructions, which are allowed to be executed only by the operating system.
   - General instructions, which can be executed by any program.
2. An interface consisting of **system calls** as offered by am operating system.
3. An interface consisting of library calls, generally forming what is known as the **application programming interface (API)**.

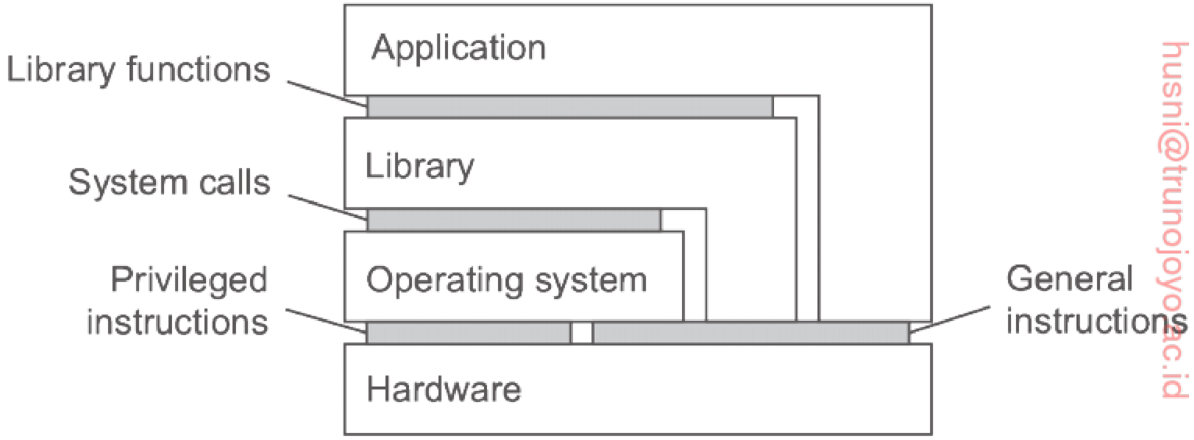

Figure 3.7: Various interfaces offered by computer systems.

Virtualization can take place in 2 different ways :

1. We can build a runtime system that essentially provides an abstract instruction set that is to be used for executing applications. Instructions can interpreted (JRE), but could also be emulated (running Windows app on unix systems) which will also have to mimic the behaviour of system calls. This is called a **process virtual machine**.

2. Provide a system that is implemented as a layer shielding the original hardware, but offering the complete instruction set of that same. This leads to what is known as **native virtual machine monitor** Called native because it is implemented directly on top of the underlying hardware. Note that the interface offered by a virtual machine can be offered *simultaneously* to different programs. As a result, it is now possible to have multiple, different **guest OS** run independently and concurrently on the same platform.
   A native VM monitor will have to provide and regulate access to various resources, like external storage and networks. Like any OS, it will have to implement device drivers for those resources. Rather than doing this anew, a **hosted virtual machine monitor** will run on top of a trusted **host OS**. It will have to given special priveledges rather than running as a user-level application.

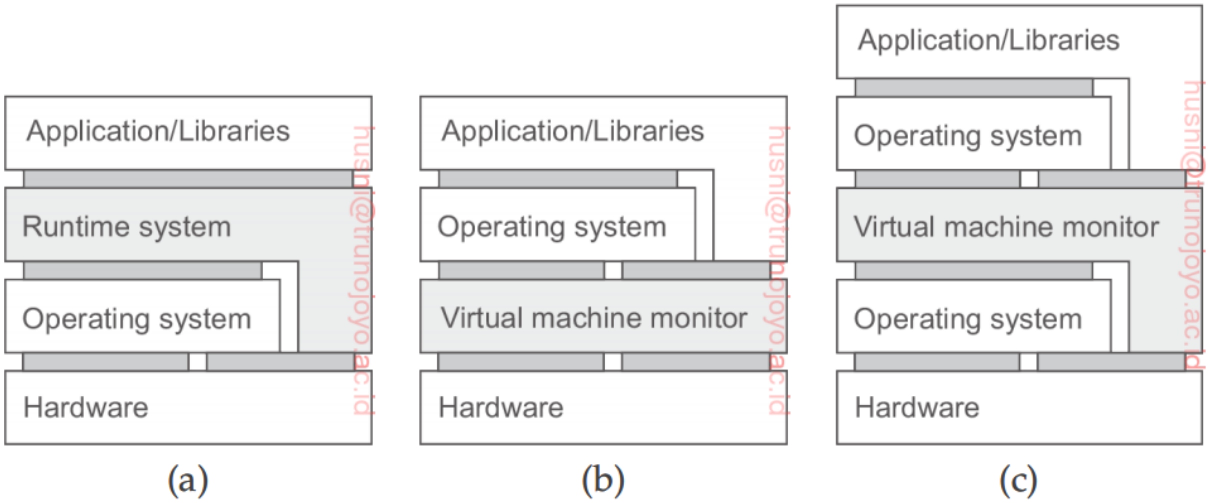

Figure 3.8: (a) A process virtual machine. (b) A native virtual machine monitor. (c) A hosted virtual machine monitor.

Application of virtualization to distributed systems

- **Infrastructure-as-a-Service (IaaS)** covering the basic infrastructure
- **Platform-as-a-service (PaaS)** covering system-level services
- **Software-as-a-service (SaaS)** containing actual applications

Infrastructure as a service (IaaS) is a type of cloud computing service that offers essential compute, storage and networking resources on demand, on a pay-as-you-go basis.

PaaS, or **Platform-as-a-Service**, is a cloud computing model that provides customers a complete cloud platform—hardware, software, and infrastructure—for developing, running, and managing applications without the cost, complexity, and inflexibility that often comes with building and maintaining that platform on-premises.

Software as a service (or **SaaS**) is a way of delivering applications over the Internet—as a service.

IaaS vs PaaS

IaaS: cloud-based services, pay-as-you-go for services such as storage, networking, and virtualization.
PaaS: hardware and software tools available over the internet.