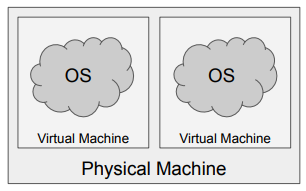
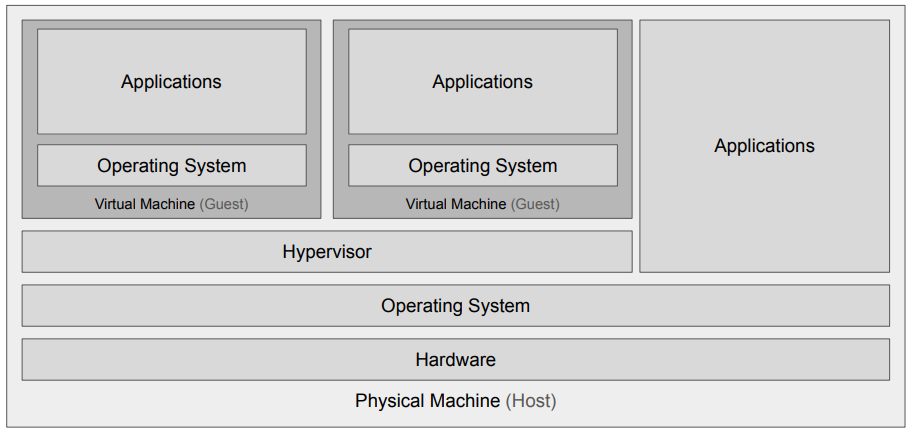
Operating Systems

Virtualisation

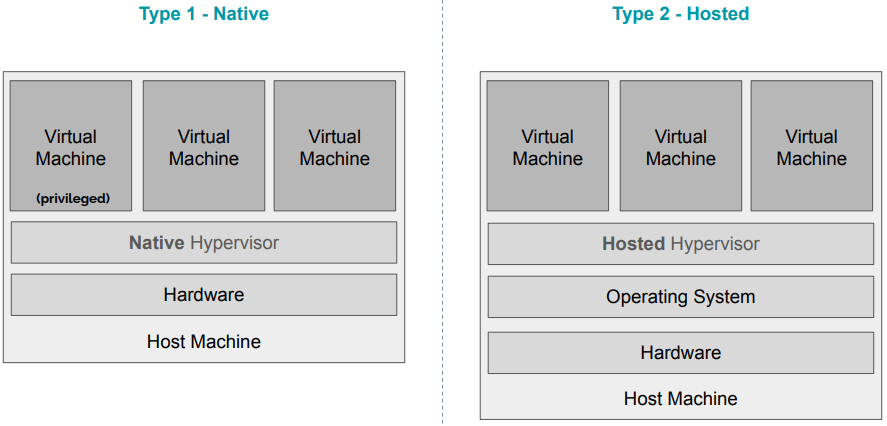
Virtualisation is the process of creating a virtual version of a physical object. In computer hardware virtualisation is the process of creating a virtual version of real hardware. This virtual hardware can be used to run a complete operating system.

Terminology

* Virtual Machine: A virtual representation of a physical machine (not to be confused with a Java Virtual Machine or the CLR (.NET))
* Virtual Machine Monitor or Hypervisor: A software application that monitors and manages running virtual machines.
* Host Machine: The physical machine that a virtual machine is running on.
* Guest Machine: The virtual machine, running on the host machine.

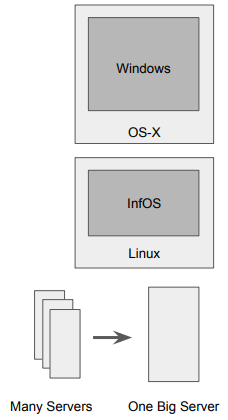
Virtual Machine Monitor (Hypervisor)

The VMM is in charge of running the virtual machines. There are two main types of VMM Type 1 (native) and type 2 (hosted) (there is also type 0 which is mainly hardware).

Type 1 or Native Hypervisors run directly on the host machine and share out resources (such as memory and devices) between the guest machines (e.g. XEN or the Oracle VM Server).

Type 2 or Hosted Hypervisors run as an application inside an operating system and support virtual machines running as individual processes (e.g. VirtualBox, Parallels Desktop and QEMU).

Uses of Virtualisation

The three main use-cases of virtualisation are:

* Personal
  + Running multiple operating systems on one host without the inconvenience of rebooting
* Technical
  + Operating System/Hardware Design
  + Kernel Debugging/Testing
  + Prototyping new architectures/architectural features
* Commercial
  + Data centre server consolidation
  + High availability/Migration

Types of Virtualisation

There are also many types of virtualisation including:

* Software Emulation
  + Maximum flexibility for virtualisation but very slow to run (high overhead)
  + Each guest instruction is emulated (can use binary translation for speed-up)
* Containers/Namespaces
  + Isolate processes/groups of processes within a single operating system e.g. Docker
* Full System or Hardware Virtualisation
  + Isolate multiple operating systems from each other within a single physical machine
* Same-architecture Virtualisation
  + Guest machine is the same architecture as the Host machine
* Cross-architecture Virtualisation
  + Guest machine has a different architecture than the host machine
  + Must use software emulation to achieve this

Popek and Goldberg Requirements for Virtualisation

There are three main properties for a virtual machine:

1. Efficiency
   * The majority of guest instructions are executed directly on the host machine
2. Resource Control
   * The virtual machine monitor must remain in control of all machine resources
3. Equivalence
   * The virtual machine must behave in a way that is indistinguishable from if it was running as a physical machine

Efficiency

“All innocuous instructions are executed by the hardware directly, with no intervention at all on the part of the control program”. This means normal guest machine instructions should be executed directly on the processor, but system instructions need to be emulated by the VMM.

Resource Control

“It must be impossible for that arbitrary program to affect the system resources, i.e. memory, available to it; the allocator of the control program is to be invoked upon any attempt.” Meaning the virtual machine should not be able to affect the host machine in any adverse way. The host machine should remain in control of all physical resources, sharing them out to guest machines.

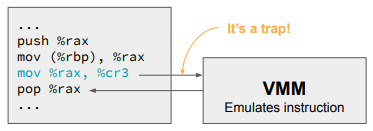
Equivalence

“Any program K executing with a control program resident, with two possible exceptions, performs in a manner indistinguishable from the case when the control program did not exist and K had whatever freedom of access to privileged instructions that the programmer had intended”. This is just a formal way of saying that the operating system running on a virtual machine should believe it is running on a physical machine, i.e. the behaviour of the virtual machine (from the guest OS’ point of view) is identical to that of the corresponding physical machine.

There are two exceptions mention, temporal latency (some instruction sequences will take longer to run) and resource availability (physical machine resources are shared between virtual machines).

Methods of Virtualisation

There are two main methods of virtualisation:

* Full software emulation
  + This isn’t permitted by Popek and Goldberg as It violates the efficiency property (though this no longer holds due to the advent of efficient binary translation)
  + This is required for cross-architecture virtualisation as guest instructions can’t execute natively on the host
* Trap-and-Emulate
  + The guest operating system runs “de-privileged”, all non-privileged instructions execute natively on the host.
  + All privileged instructions trap to the VMM
  + VMM emulates these privileged operations
  + Guest resumes execute after emulation

Virtualising x86

Originally x86 was not “classically” virtualizable as some privileged instructions didn’t “trap” and so couldn’t be emulated correctly. For solving this issue interpretation is too slow (violates efficiency) and code patching leaves traces of virtualisation (violates equivalency). Binary translation is better, but still incurs overhead. Since 2005, x86 processors have supported virtualisation in hardware, this enables trap-and-emulate style virtualisation at the hardware level so unmodified operating systems can run natively on host machines.

Hardware Acceleration for Virtualisation

Modern processors include hardware support for running virtual machines (Intel VT-X and AMB-V for x86, ARM Virtualisation Extensions for ARM processors).

Hardware extensions allow all guest instructions (including system instructions) to run natively on the processor. This works by providing an isolated view of the processor to virtual machines, the operating systems can then run directly on the processor believing they are running on physical hardware. Certain privileged operations still “trap” back to the hypervisor.

Virtual Machine Access to Resources

Virtual machines need to be given access to resources such as: memory, storage, networking, graphics etc… It’s the responsibility of the VMM to share out these resources. Access to physical memory is managed by the VMM too. For an unmodified operating system, expecting a “real” storage device (e.g. a hard disk), the VMM must provide an emulation of that device. Some devices may be passed straight through to the virtual machine though (e.g. dedicated network cards).

Paravirtualisation

With paravirtualisation the guest operating systems are aware they are being virtualised and co-operate with the hypervisor to enable increased memory and device performance. They no longer “trap-and-emulate” but instead request privileged operations directly from the hypervisor. They can co-operate with the hypervisor so that host memory can be more efficiently distributed. Instead of providing an emulated storage device, the hypervisor can provide a paravirtualised implementation. Paravirtualisation is typically used in data centres for large-scale virtualisation.