Cloud Computing (ECS781P)

Week 3: A bit more on Virtualization, Containers

Dr Arman Khouzani, Dr Felix Cuadrado

Electronic Engineering and Computer Science
Queen Mary University of London

January 26, 2018

Queen Mary
University of London

Virtualisation: Recap

- The concept of virtualisation:
  - Managing the underlying (physical) resources to create an abstracted pool of the virtualised (aka *logical*) resource, which can be divided up into standard units and allocated by the upper layers.
  - "resources" are anything required for computing: processor, memory, storage, networking, I/O, etc.
  - "Management" of the physical resources involve isolation and prevention of conflict when the virtual resources are used by different guests (multi-tenancy) while the physical resources is indeed shared by all of them.
  - Usage of an instance of a virtual resource should not affect another instances of the virtual resource.
  - "Abstraction" in part implies that the users of virtual resource should no longer have access to the physical resource.

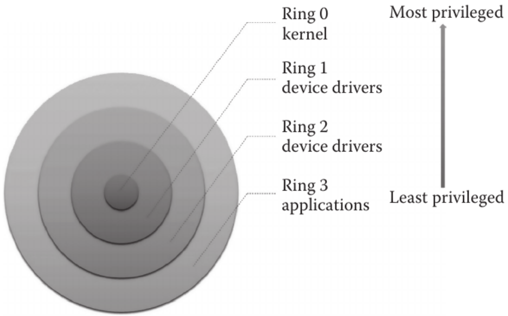
Virtualisation: Recap

- Different computing resources that can be virtualised:
  - Processor Virtualisation (creating a pool of virtual CPU units to be assigned to VMs)
  - Main Memory Virtualisation (creating a pool of virtual RAM units to be allocated to VMs)
  - Storage Virtualisation (creating a pool of virtual storage disks, i.e., logical storage, to be allocated to VMs, or for maintaining backups/replica of the VM virtual storages for improved availability, etc.)
  - Network virtualisation, . . .

Protecting access to the physical resources

Protection rings in processor architecture:

- ▷ a hierarchically arranged architecture of privilege levels used to isolate the OS from user applications.



Typically, the hypervisor runs at *ring 0* (most privilege) and guest OSs run at a lower privileged level.

Approaches to Virtualisation

Different approaches to virtualisation:

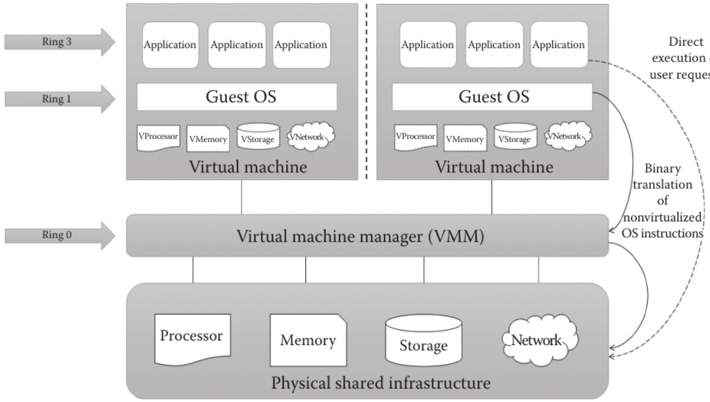
1. **Full virtualisation**:
2. **Paravirtualisation**:
3. **Hardware-assisted virtualisation**

Full Virtualisation

Full Virtualisation: the virtual machine manager (VMM) fully abstracts the underlying infrastructure and decouples the guest OS from it.

- Hence the guest-OS's kernel can be unmodified.
- uses techniques like:
  - *binary translation:* the sequences of instructions are translated from a source instruction set (of the guest OS) to the target instruction set (of the host OS/hardware)
    - This can be done for the entire instruction set, allowing full emulation of a processor architecture over another processor architecture, e.g. KEMU.
    - or done only for a subset of the code that needs privileged execution (Ring 0) such as kernel-mode, e.g. VMware virtualization solutions.
  - *direct execution:* user application requests that can directly execute on the host CPU.

Full Virtualisation



Full Virtualisation

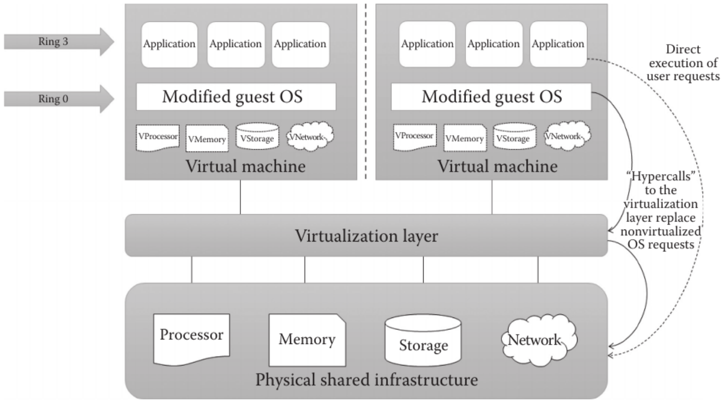
- **Pros**
  - portability of application code and OSs
  - best isolation and security for the VMs
  - easy to install and use and does not require any change in the guest OS.
- **Cons**
  - Overhead introduced by “binary translation” reduces the overall system performance.
  - There is a need for correct combination of hardware and software (the support of unmodified guest OSs is limited to those that can run in the host CPU, unless you are using full emulation, which has a higher overhead)

Paravirtualisation

Paravirtualization, a.k.a., *partial virtualization* or *OS-assisted virtualization*: provides partial emulation of the underlying infrastructure.

- ▷ In full virtualization, the guest OS is used without any modification. But in paravirtualization, the guest OS needs to be modified to replace nonvirtualizable instructions with *hypercalls*.
- ▷ a hypercall to a hypervisor is the equivalent of a syscall to an OS kernel
- ▷ the guest OS is at privileged level, so it can communicate directly using hypercalls with the virtualization layer without any translation.
- ▷ examples: Denali, Xen, Hyper-V

Paravirtualisation



Paravirtualisation

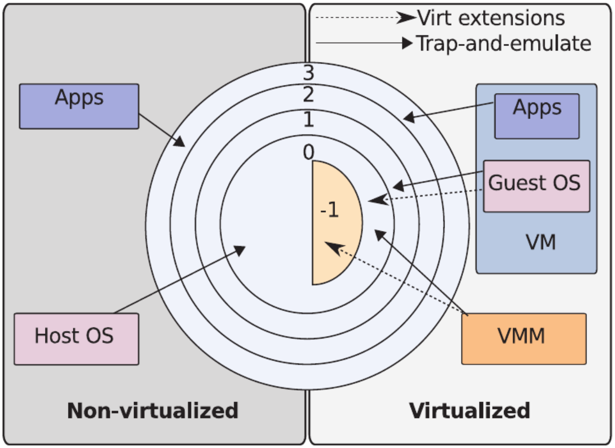
- **Pro:**
  - low virtualisation overhead: eliminates the additional overhead of binary translation and hence improves the overall system efficiency and performance.
- **Con:**
  - the kernel of the guest OSs needs to be modified in order to make use of the hypercalls
  - so unlike full virtualisation, the modified guest OS cannot be migrated to run on physical hardware.
  - it also raises the problem of backward compatibility and hypervisor dependence for the VMs.

Hardware-Assisted Virtualisation

As of 2006, processor vendors such as AMD and Intel introduced virtualisation extensions to their line of products.

- it implements a Ring with a higher privileged mode in the processor architecture
- The CPU extensions for virtualization support allows executing unmodified guest OSs in Ring 0 (non-root mode) and the VMM or Hypervisor in Ring -1 (root mode)
- no need for binary translation or Para-virtualisation
- Examples: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, Hyper-V, and VMware products

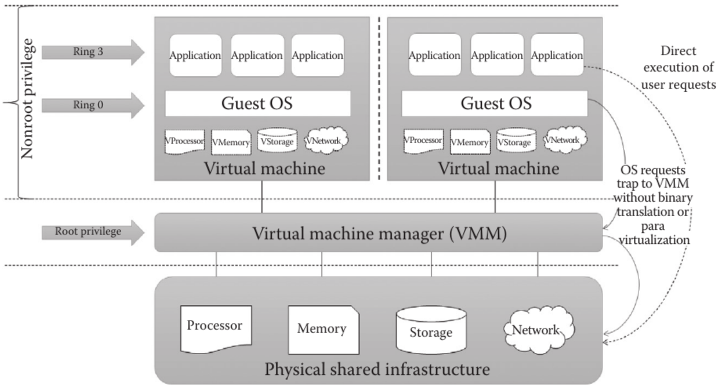
Protecting access to the physical resources



1

[1] Ref: F. Rodrguez-Haroa, *et al.* “A summary of virtualization Techniques”. Procedia Technology 3 (2012) 267–272

Hardware-Assisted Virtualisation



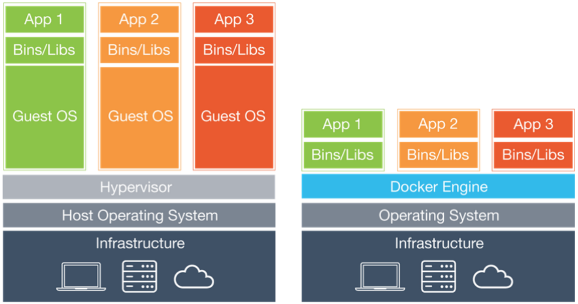
Hardware-Assisted Virtualisation

- **Pro:**
  - low virtualisation overhead: no binary translation of full virtualisation.
  - no need for guest OS modification as in paravirtualisation.
- **Con:**
  - Only new-generation processors have these capabilities
  - More number of VM traps result in high CPU overhead, limited scalability, and less efficiency in server consolidation

Application Containers

- Application containerization, or simply, containerization is closely related to virtual machines, but has very important distinctions.
- In VM, the entire OS is emulated (including all the functionalities of the kernel).
- if our objective is to run applications that will run on any hardware and OS, then spinning an entire virtual machine looks like a rather waste of the host resources.
- Solution: Application containerisation (e.g `Docker`, `LXC`, `runC`, etc

Application Containers



Although the docker engine creates a virtual platform for the apps to run on, it re-uses a lot of the resources of the host Kernel, as opposed to creating a separate guest Kernel per each app.