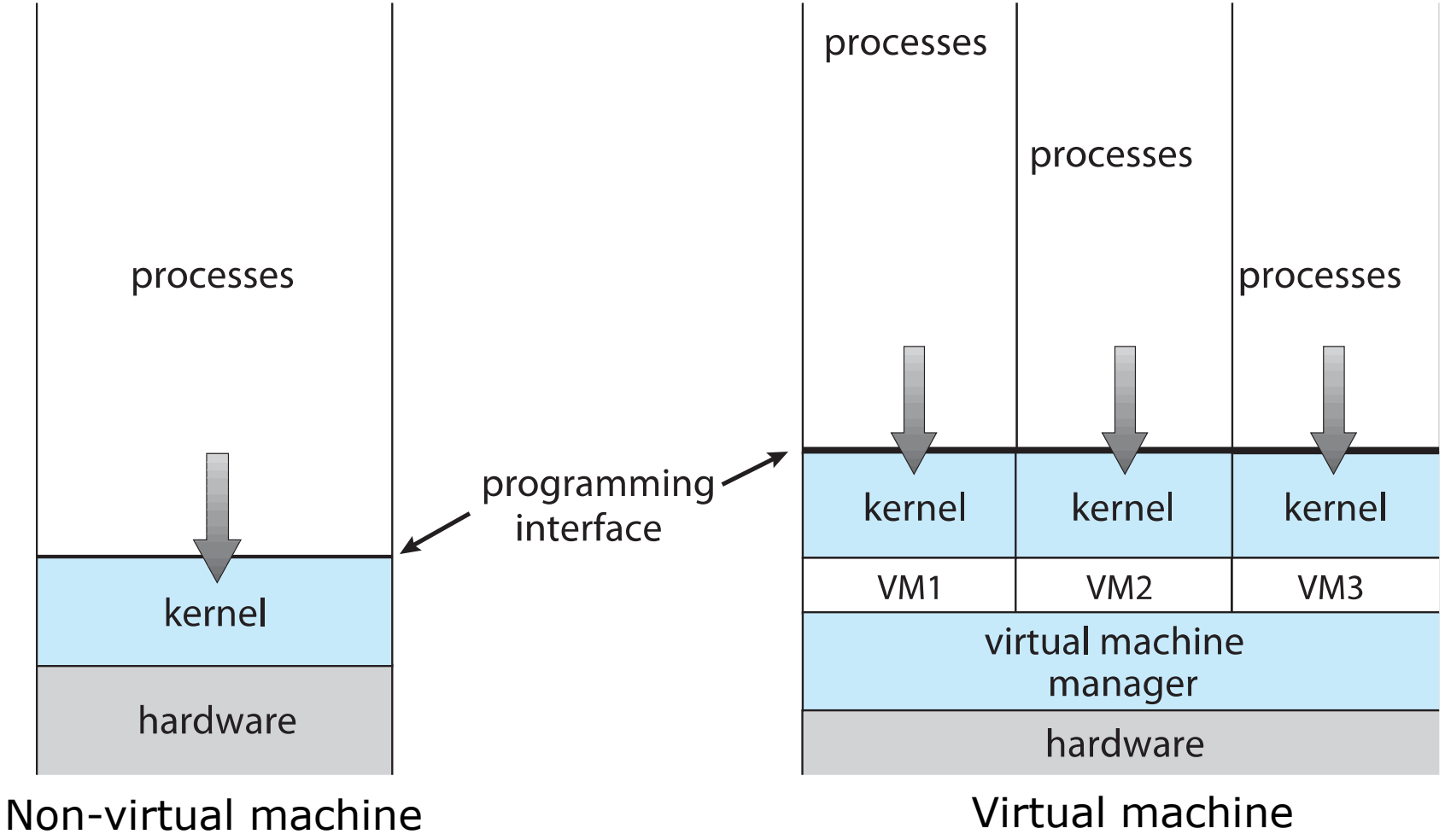
Virtualisation and the Cloud

Operating Systems

Virtualisation

- Fundamental idea – abstract hardware of a single computer into several different execution environments
  - Similar to layered approach, but layer creates virtual system (**virtual machine**, or **VM**) on which operation systems or applications can run
- Single physical machine can run multiple operating systems concurrently, each in its own virtual machine

Virtualisation



Virtualisation

- Several components:
- **Host**
  - underlying hardware system
- **Virtual machine manager** (**VMM**) or **hypervisor**
  - creates and runs virtual machines by providing interface that is ***identical*** to the host
- **Guest**
  - process provided with virtual copy of the host

History

- First appeared in IBM mainframes in 1972
- Allowed multiple users to share a batch-oriented system
- Formal definition of virtualization helped move it beyond IBM
  1. A VMM provides an environment for programs that is essentially identical to the original machine
  2. Programs running within that environment show only minor performance decreases
  3. The VMM is in complete control of system resources
- In late 1990s, Intel CPUs became fast enough for researchers to try virtualizing on general purpose PCs
  - **Xen** and **VMware** created technologies, still used today
  - Virtualization has expanded to many OSes, CPUs, VMMs

Benefits and Features

- Host system protected from VMs, VMs protected from each other
  - Sharing is provided though via shared file system volume, network communication
- Freeze, **suspend**, run VM
  - Then can move or copy somewhere else and **resume**
  - Snapshot of a given state, able to restore back to that state
    - Some VMMs allow multiple snapshots per VM
  - **Clone** by creating copy and running both original and copy

Benefits and Features

- Run multiple, different OSes on a single machine
  - **Consolidation**, app dev, etc.
- **Templating**
  - create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- **Live migration**
  - move a running VM from one host to another!
  - No interruption of user access

Benefits and Features

- All those features taken together -> **cloud computing**
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops
- Great for OS research, better system development efficiency

Types of VMs and Implementations

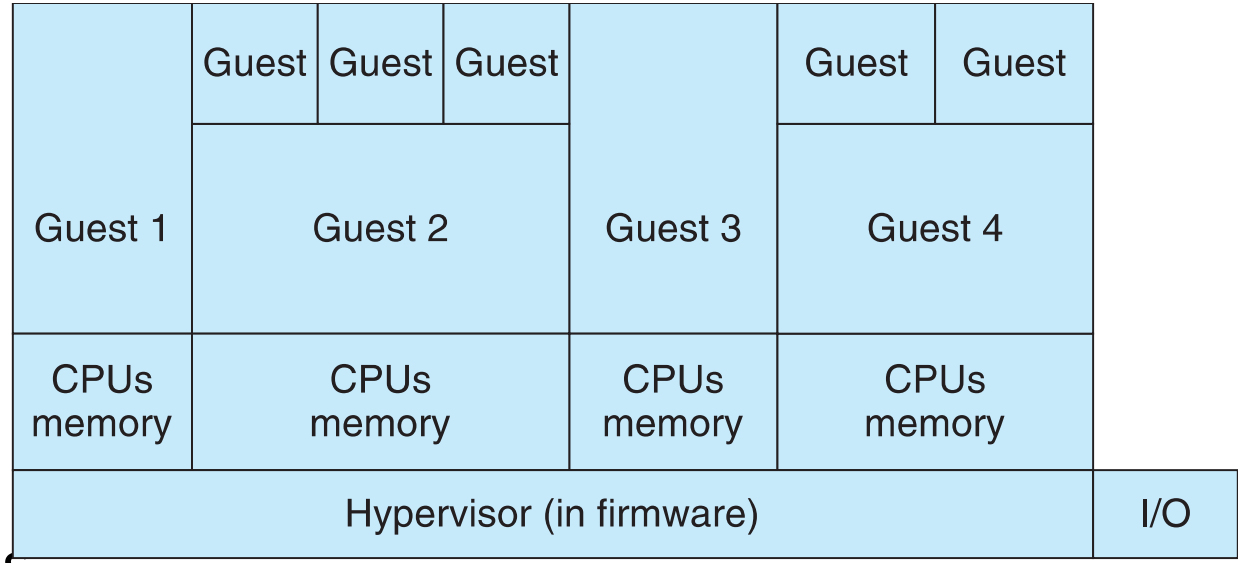
- Many variations as well as HW details
  - Assume VMMs take advantage of HW features
- Steps simpler, faster than with a physical machine install
  - Can lead to **virtual machine sprawl** with lots of VMs, history and state difficult to track

Types of VMs and Implementations

- Whatever the type, a VM has a lifecycle
  - Created by VMM
  - Resources assigned to it (number of cores, amount of memory, networking details, storage details)
  - In type 0 hypervisor, resources usually dedicated
  - Other types dedicate or share resources, or a mix
  - When no longer needed, VM can be deleted, freeing resources

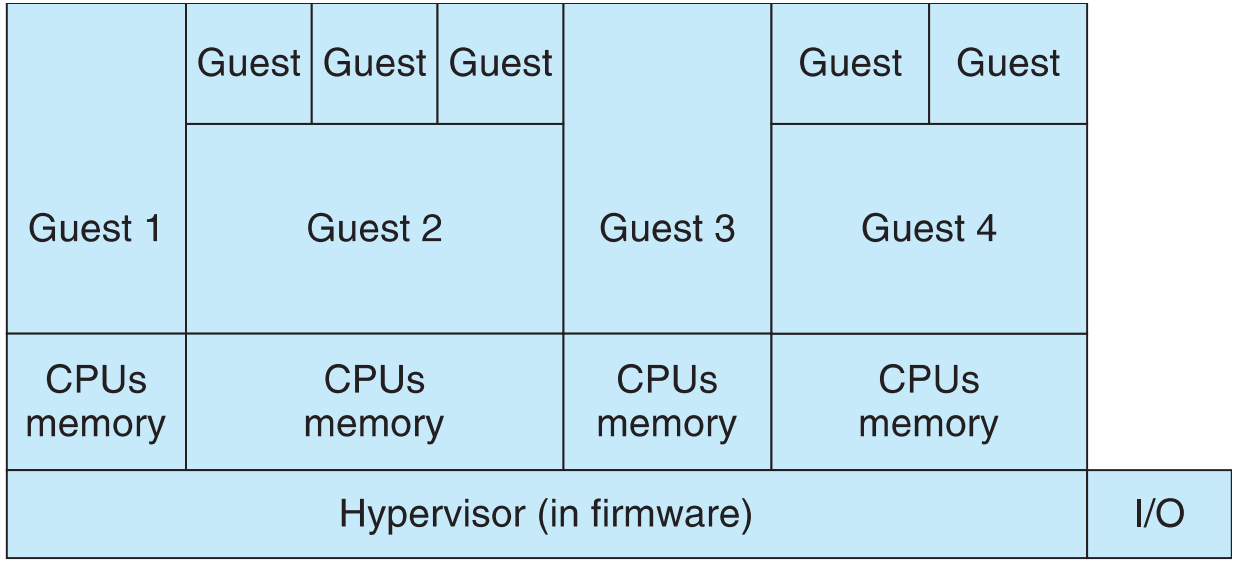
Types of VMs – Type 0 Hypervisor

- Oldest VM idea
- From different HW manufacturers
- VMM is a HW feature
- Implemented by firmware
- OS need to nothing special
- Smaller feature set than other types
- “Partitions”, “Domains”
- Each guest has dedicated HW



Types of VMs – Type 0 Hypervisor

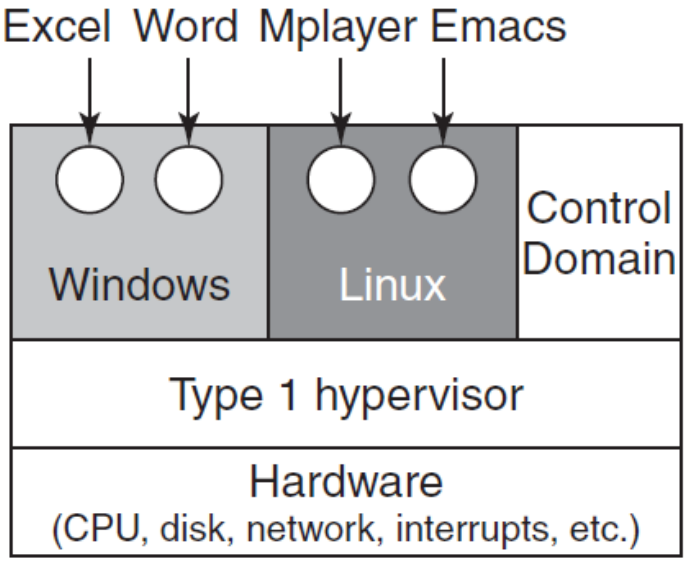
- I/O a challenge as difficult to have enough devices, controllers to dedicate to each guest
- Sometimes VMM implements a **control partition** running daemons that other guests communicate with for shared I/O



- Can provide virtualization-within-virtualization
  - Guest itself can be a VMM with guests, other types have difficulty doing this

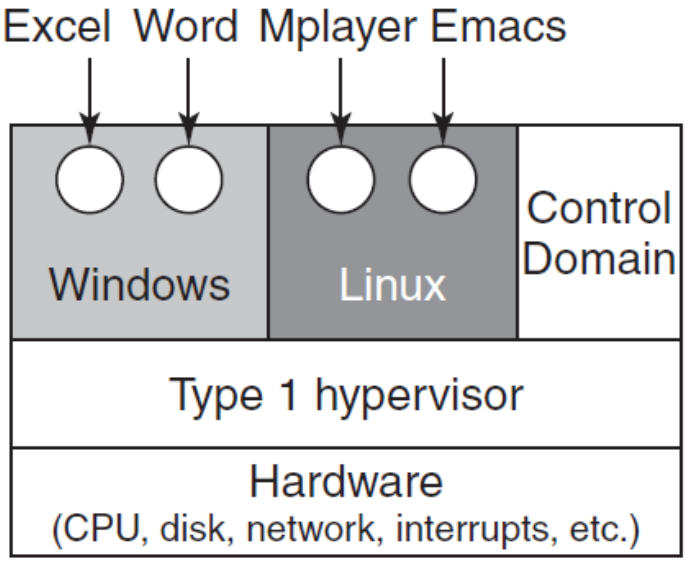
Types of VMs – Type 1 Hypervisor

- Special purpose operating systems that run natively on HW
- Consolidation of multiple OSes and apps onto less HW
- Move guests between systems to balance performance
- Commonly found in datacenters
- Datacenter managers control and manage OSes in new, sophisticated ways by controlling the Type 1 hypervisor
- Snapshots and cloning



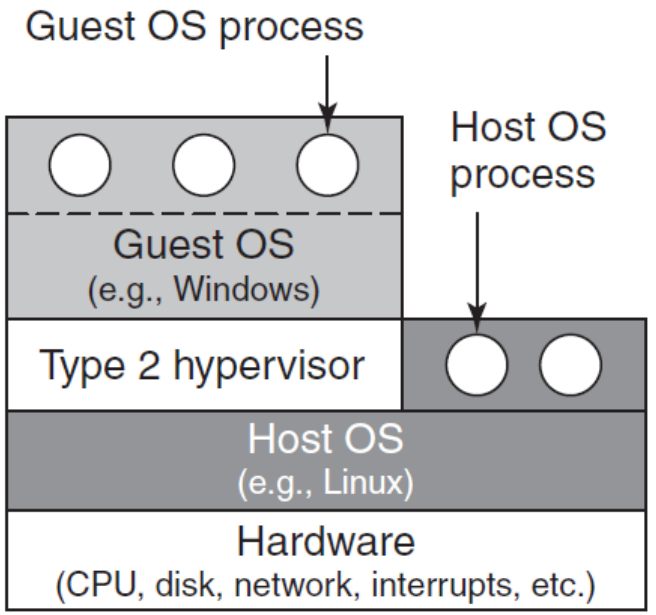
Types of VMs – Type 1 Hypervisor

- Run in kernel mode
- Can run on top of Type 0s but not on other Type 1s
- Guests generally don't know they are running in a VM
- Rather than providing system call interface, create run and manage guest OSes
- Implement device drivers for host HW because no other component can
- Also provide other traditional OS services like CPU and memory management



Types of VMs – Type 2 Hypervisor

- Very little OS involvement in virtualization
- VMM is simply another process, run and managed by host
  - Even the host doesn't know they are a VMM running guests
- Tend to have poorer overall performance because can't take advantage of some HW features
- But also a benefit because require no changes to host OS
  - Student could have Type 2 hypervisor on native host, run multiple guests, all on standard host OS such as Windows, Linux, MacOS



OS Component – CPU Scheduling

- Even single-CPU systems act like multiprocessor ones when virtualized
  - One or more virtual CPUs per guest
- Generally VMM has one or more physical CPUs and number of threads to run on them
  - Guests configured with certain number of VCPUs
    - Can be adjusted throughout life of VM
  - When enough CPUs for all guests -> VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
  - Usually not enough CPUs -> CPU overcommitment
    - VMM can use standard scheduling algorithms to put threads on CPUs
    - Some add fairness aspect

OS Component – Memory Management

- Can suffer from oversubscription, requires extra management efficiency from VMM
- For example, VMware ESX guests have a configured amount of physical memory, then ESX uses 3 methods of memory management
  1. Double-paging, in which the guest page table indicates a page is in a physical frame but the VMM moves some of those pages to backing store
  2. Install a **pseudo-device driver** in each guest (it looks like a device driver to the guest kernel but really just adds kernel-mode code to the guest)
     - **Balloon** memory manager communicates with VMM and is told to allocate or deallocate memory to decrease or increase physical memory use of guest, causing guest OS to free or have more memory available
  3. Deduplication by VMM determining if same page loaded more than once, memory mapping the same page into multiple guests

OS Component – I/O

- Easier for VMMs to integrate with guests because I/O has lots of variation
  - Already somewhat segregated / flexible via device drivers
  - VMM can provide new devices and device drivers
- But overall I/O is complicated for VMMs
  - Many short paths for I/O in standard OSes for improved performance
  - Less hypervisor needs to do for I/O for guests, the better
  - Possibilities include direct device access, DMA pass-through, direct interrupt delivery
    - Again, HW support needed for these
- Networking also complex as VMM and guests all need network access
  - VMM can **bridge** guest to network (allowing direct access)
  - And / or provide **network address translation** (**NAT**)
    - NAT address local to machine on which guest is running, VMM provides address translation to guest to hide its address

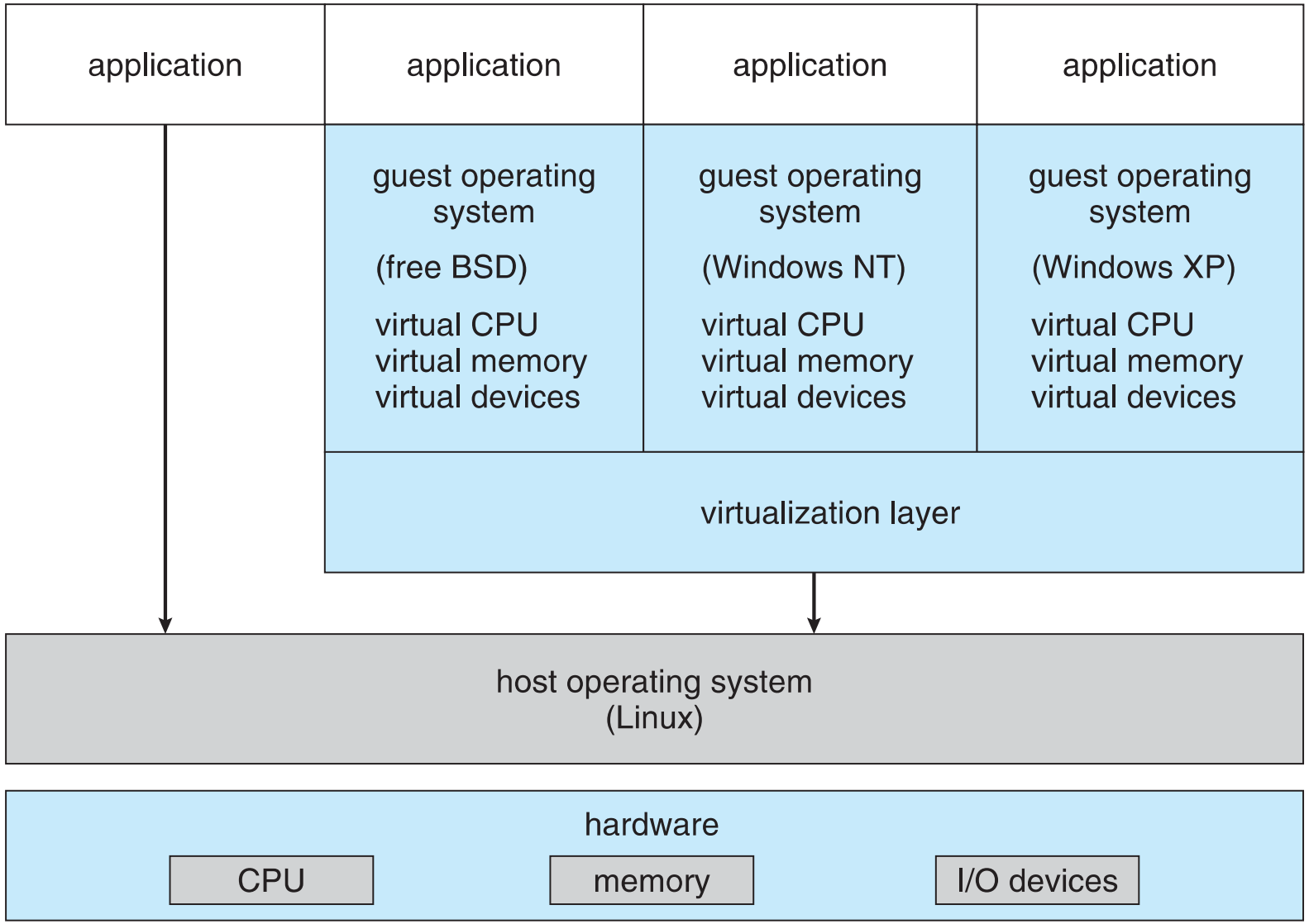
OS Component – Storage Management

- Both boot disk and general data access need be provided by VMM
- Need to support potentially dozens of guests per VMM (so standard disk partitioning not sufficient)
- Type 1 – storage guest root disks and config information within file system provided by VMM as a disk image
- Type 2 – store as files in file system provided by host OS

OS Component – Storage Management

- Duplicate file
  - create new guest
- Move file to another system
  - move guest
- **Physical-to-virtual (P-to-V)**
  - convert native disk blocks into VMM format
- **Virtual-to-physical (V-to-P)**
  - convert from virtual format to native or disk format
- VMM also needs to provide access to network attached storage, other disk images, disk partitions, disks, etc.

Example - VMware



Example - VMware

- VMware Workstation runs on x86, provides VMM for guests
- Runs as application on other native, installed host operating system -> Type 2
- Lots of guests possible, including Windows, Linux, etc all runnable concurrently (as resources allow)
- Virtualization layer abstracts underlying HW, providing guest with is own virtual CPUs, memory, disk drives, network interfaces, etc
- Physical disks can be provided to guests, or virtual physical disks (just files within host file system)

Virtualization Security Requirements

- Scenario: A client uses the service of a cloud computing company to build a remote VM
  - A secure network interface
  - A secure secondary storage
  - A secure run-time environment
    - Build, save, restore, destroy

Virtualization Security Requirements

- A secure run-time environment is the most fundamental
- The first two problems already have solutions:
  - Network interface: Transport layer security (TLS)
  - Secondary storage: Network file system (NFS)
- The security mechanism in the first two rely on a secure run-time environment
  - All the cryptographic algorithms and security protocols reside in the run-time environment

Clouds

- Virtualisation technology played a crucial role in the rise of cloud computing.
- Cloud providers typically offer different categories of resources, such as ‘‘big machines’’ versus ‘‘little machines,’’ etc.
- Can be private (to an organisation), or public (to anyone)
- Can access to physical hardware, or virtualise their environments
- Can be only bare machines, or offered with ready to use software

Characteristics of Clouds

- Five essential characteristics of clouds according to the National Institute of Standards and Technology:

1. **On-demand self-service.** Users should be able to provision resources automatically, without requiring human interaction.
2. **Broad network access.** All these resources should be available over the network via standard mechanisms so that heterogeneous devices can make use of them.

Characteristics of Clouds

3. **Resource pooling.** The computing resource owned by the provider should be pooled to serve multiple users and with the ability to assign and reassign resources dynamically. The users generally do not even know the exact location of ‘‘their’’ resources or even which country they are located in.
4. **Rapid elasticity.** It should be possible to acquire and release resources elastically, perhaps even automatically, to scale immediately with the users’ demands.
5. **Measured service.** The cloud provider meters the resources used in a way that matches the type of service agreed upon.

Clouds as a Service

- **SAAS** (**Software As A Service)**
  - offers access to specific software, such as Microsoft Office 365 or Google Apps
- **PAAS** (**Platform As A Service)**
  - delivers an environment that includes things such as a specific OS, database, Web server, and so on
- **IAAS** (**Infrastructure As A Service**)
  - offer direct access to a virtual machine, which the user can use in any way
  - example of an IAAS cloud is Amazon EC2

Clouds as a Service

- Clouds can transform the way companies do computing.
- Benefits from economy of scale,
  - Consolidating the computing resources in a small number of places
  - Conveniently located near a power source and cheap cooling
- No need to worry so much about managing the IT infrastructure, backups, maintenance, depreciation, scalability, reliability, performance, and perhaps security.

Emerging Challengers with Clouds

- Can you really trust a cloud provider to keep your sensitive data safe?
- Will a competitor running on the same infrastructure be able to infer information you wanted to keep private?
- What law(s) apply to your data (for instance, if the cloud provider is from the United States, is your data subject to the PATRIOT Act, even if your company is in Europe or Asia)?
- Once you store all your data in cloud X, will you be able to get them out again, or will you be tied to that cloud and its provider forever, something known as **vendor lock-in**?