Say you have a house to rent…

- What does the tenant want?
  - An independent house ☺
- What can you give?

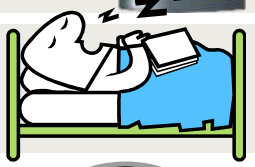
What does a tenant look for?

- Is it affordable?

- Is there enough space?

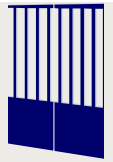
- Is it safe from outsiders?
  - Is it safe from other tenants? Locks, shades, ..

- Will I not be disturbed by tenants?



- Is power billed separately?

- Can I get a separate main entrance?
  - Or at least make sure I don't have to fight crowds?



- Do I have to share the verandah!!?

Say you have a computer to rent…

- What does the “tenant” want?
  - Their own computer ☺
- What can you give?
  - And how?

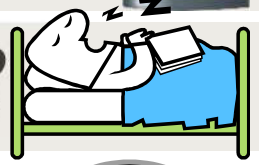
What does a tenant look for?

- Is it affordable to rent?

- Is there enough CPU/memory?

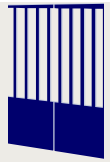
- Is it safe from the N/W?
  - Is it safe from other users? Mem/Code Leaks..

- Will their application use affect me?



- Can I pay for what *I* use?

- Can I get my own N/W connection?
  - Or at least have a reserved bandwidth?



- What do you mean I share the disk!!?

Why Virtualize?

- Share same hardware among independent users
  - Degrees of HW parallelism increasing
- Reduce HW foot print thru' consolidation
  - Eases management, energy usage
- Sandbox/migrate applications
  - Flexible allocation, utilization
- Decouple applications from underlying HW
  - Allows HW upgrades without impact on OS image

Virtualization raises the Abstraction

?

- Similar to *Virtual Memory* to access larger address space
  - *Physical memory* mapping is hidden by OS using *paging*
- Similar to hardware emulators
  - Allow code on one arch to run on a different
- Physical devices -> Virtual Devices
  - CPU, Memory, VHD, NIC
- Now worry/not be aware of physical hardware details

Virtualization Requirements*

- Efficiency Property
  - *All innocuous instructions are executed by hardware*
- Resource Control Property
  - *It must be impossible for programs to directly affect system resources*
- Equivalence Property
  - *A program with a VMM performs in a manner indistinguishable from another without*
  - *Except: (1) Timing, (2) Resource Availability*

* Formal Requirements for Virtualizable 3rd Generation Architectures, Popek & Goldberg, *CACM*, 1974

Types of Virtualization

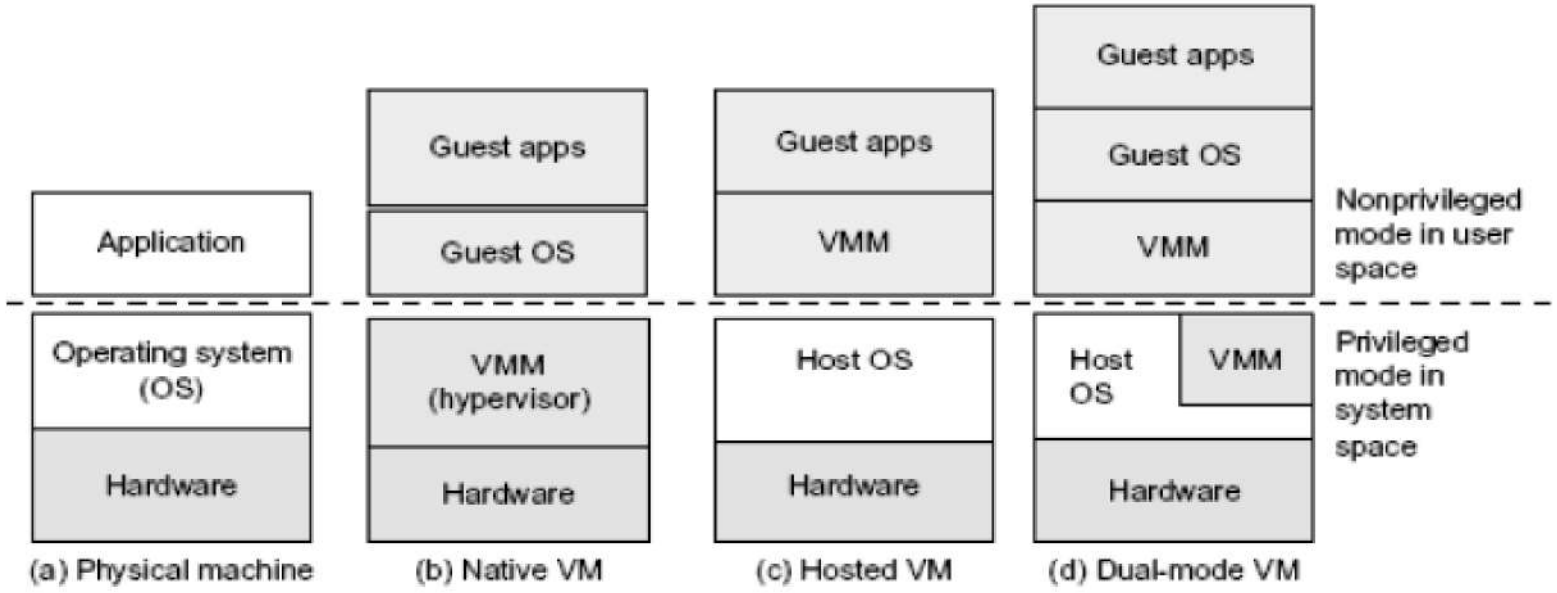


FIGURE 1.12

Three VM architectures in (b), (c), and (d), compared with the traditional physical machine shown in (a).

- Virtual Machine Manager/Virtual Machine Monitor/Hypervisor … *a Caretaker*
- Native (Hyper-V, ~KVM), Hosted (Xen)

Fig. from "Distributed & Cloud Computing, Hwang, et al, MKP; *Courtesy of Kai Hwang, USC*

Types of Virtualization*

- Full Virtualization
  - *Unmodified* Guest OS
  - VMM *binary translates kernel* to *trap* privileged calls
  - Software emulation
  - VMWare Server, Apple Parallels
- Pros
  - Guest OS not modified
  - No HW support required
- Cons
  - Binary translation costly, difficult

?

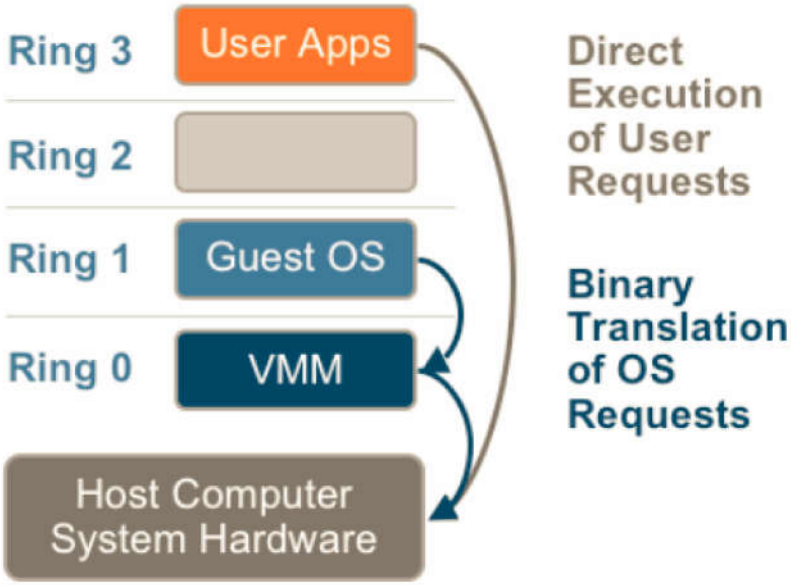


Figure 5 – The binary translation approach to x86 virtualization

*Understanding Full Virtualization, Paravirtualization and Hardware Assist, VMWare, Tech Report WP-028-PRD-01-01, 2007

Types of Virtualization

- Para-virtualization
  - Guest OS *modified* to make “hyper-calls” for privileged instructions
  - Xen in para mode
- Pro
  - (Mostly) faster & easier than bin. translation
- Con
  - Guest OS modified… Legacy, maintenance

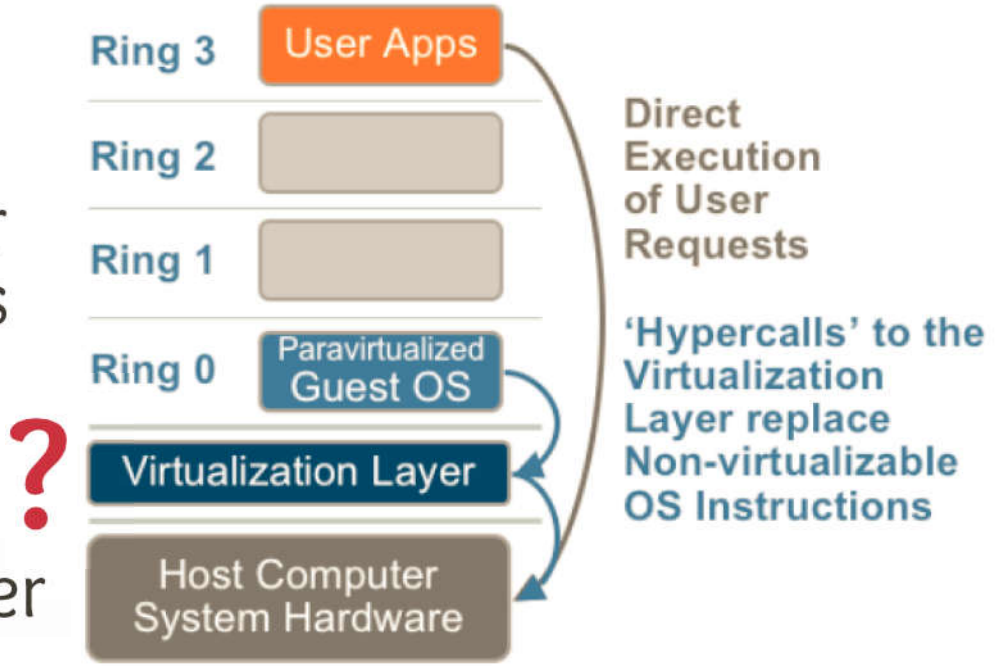


Figure 6 – The Paravirtualization approach to x86 Virtualization

Types of Virtualization

- H/W Assisted Virtualization
  - *Unmodified* Guest OS
  - CPU traps & calls VMM for privileged calls
  - CPU support in Intel VTx, AMD-V
  - Xen HVM, Hyper-V, KVM
- Pros
  - Faster to execute
  - Easier management
- Cons
  - Requires CPU support

?

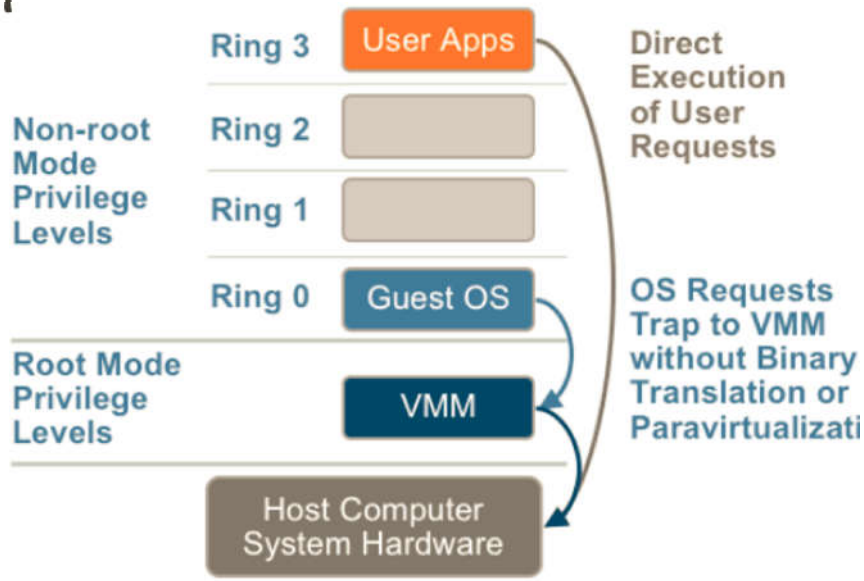


Figure 7 – The hardware assist approach to x86 virtualization

Types of Virtualization

- OS Level Virtualization
  - OS provides containers for isolation
  - Retains host OS image
  - Linux `chroot`, OpenVZ
- Pros
  - Faster to boot
  - Fewer images to maintain
  - No CPU support required
- Cons
  - No guest OS, limited distros

?

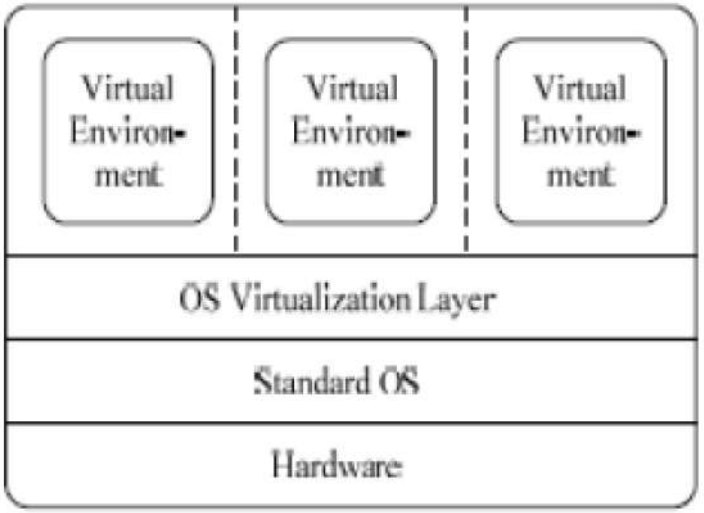
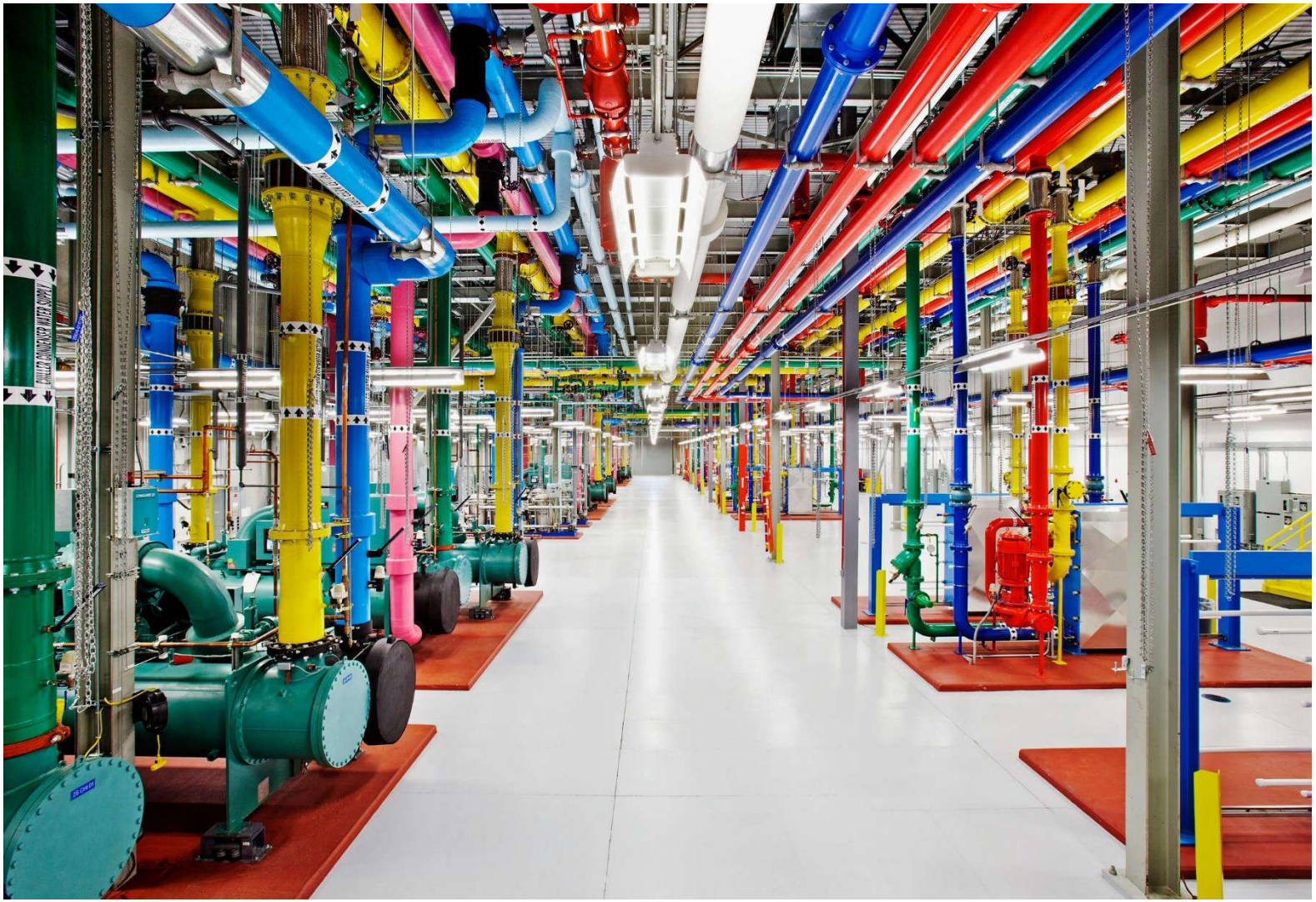


Fig. from "Distributed & Cloud Computing, Hwang, et al, MKP; *Courtesy of Kai Hwang, USC*

Centrality comes a full circle

- Mainframes -> Personal Computers -> Independent Servers -> Enterprise Servers -> Data Centres
- Data centres (More in Lecture #5)
  - Consolidate hardware, infrastructure, energy usage
  - Ease management, automation, physical security
  - Allow transparent HW improvements
- Started as enterprise-scale data centres…

Cisco’s Data Center in Texas



Google's Data Center in Georgia

Microsoft's Data Center in Ireland

NSA’s Data Center in Utah

How does this all relate to Cloud Computing?

- Rent out spare capacity in Enterprise Data Centres
  - Amazon AWS, etc.
- Build Data Centres where HW can be outsourced
  - Rackspace, etc.

A Colony to rent

Levels of Abstractions: IaaS, SaaS, PaaS *(More in Lecture #3)*

- Infrastructure as a Service (IaaS)
  - Rent out virtual machines
  - E.g. Amazon AWS's Elastic Computing Cloud
- Platform as a Service (PaaS)
  - Programming APIs that can be scaled
  - E.g. Microsoft Azure's Workers, Google App Engine, AWS Elastic MapReduce
- Software as a Service (SaaS)
  - End use applications that can be composed & scaled
  - E.g. GMail, Office365, DropBox