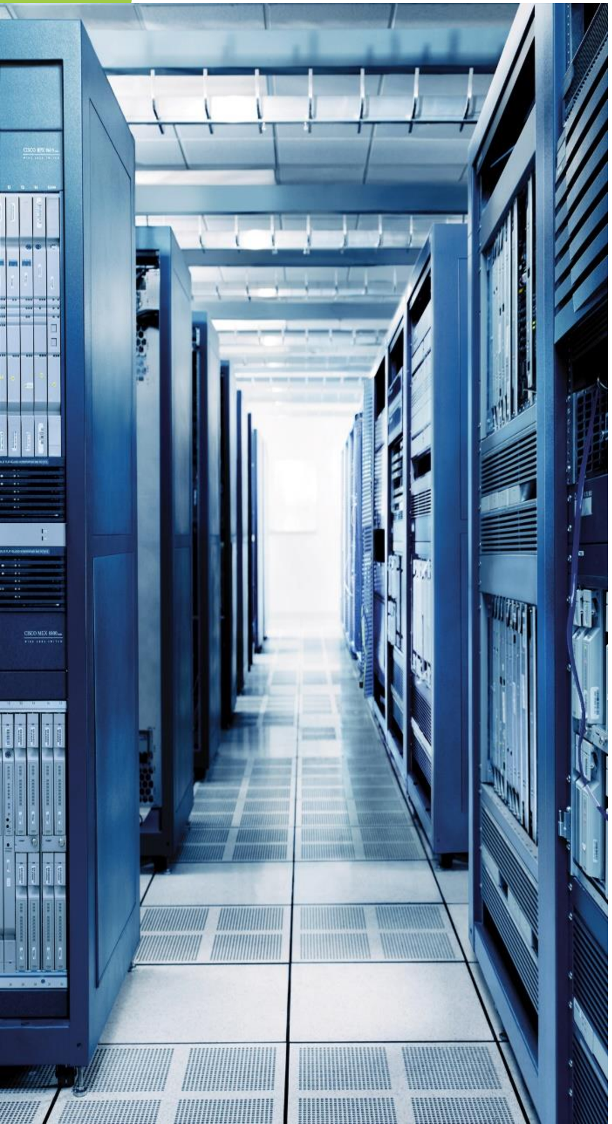
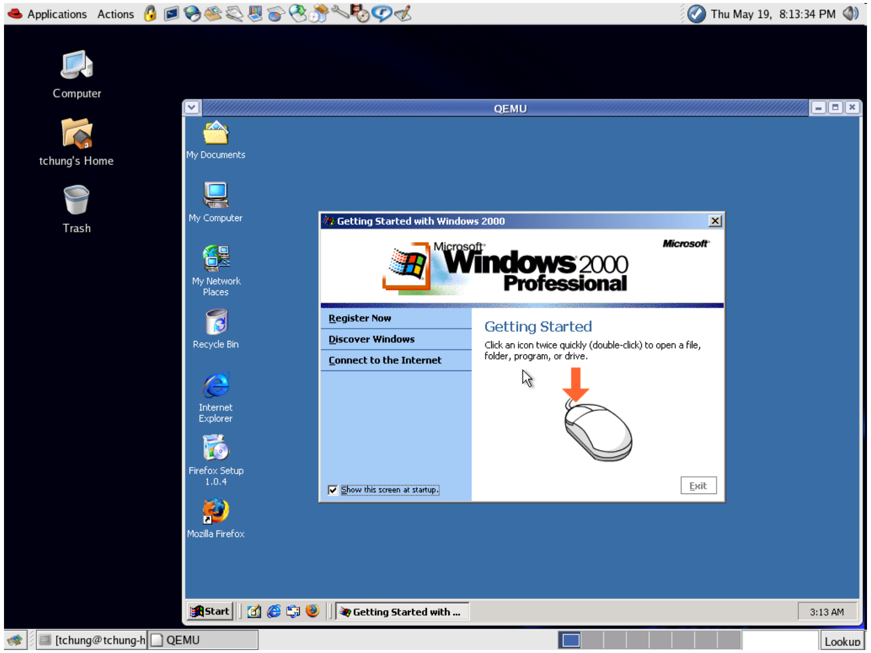


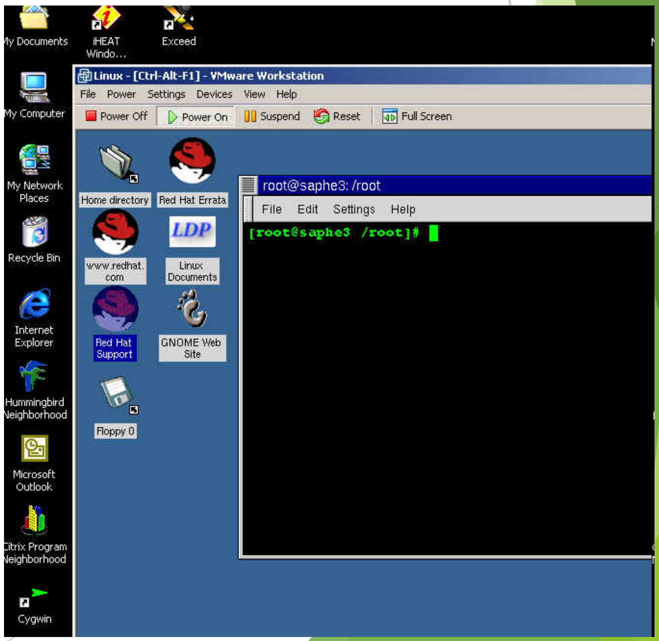
Virtualization Technology Introduction

What is virtualization?

- Virtualization is way to run **multiple operating systems** and **user applications** on the same hardware
  - E.g., run both Windows and Linux on the same laptop
- How is it different from **dual-boot**?
  - Both OSes run **simultaneously**
- The OSes are completely **isolated** from each other



Click an icon twice quickly (double-click) to open a file, folder, program, or drive.



```
[root@saphe3 /root]#
```

Uses of virtualization

- Server consolidation
  - Run a **web server** and a **mail server** on the **same physical server**
- Easier development
  - Develop critical **operating system components** (file system, disk driver) without affecting **computer stability**
- QA
  - Testing a network product (e.g., a firewall) may require **tens of computers**
  - Try testing thoroughly a product at each pre-release milestone… and have a straight face when your boss shows you the **electricity bill**
- Cloud computing

What’s new in that? We’ve been doing it for decades!

- Indeed – an OS provides **isolation** between **processes**
  - Each has it’s own **virtual memory**
  - Controlled access to **I/O devices** (disk, network) via system calls
  - Process **scheduler** to decide which process runs on which CPU core
- So what’s the hype about?
- Try running Microsoft Exchange requiring **Windows and** your internal warehouse mgmt. application requiring **Linux simultaneously** on the same server!
- Or better yet, try to persuade **competing companies** to run their **processes side-by-side** in Amazon’s **cloud** (had it not been virtualized)
- Psychological effect – what sounds better?
  - You’re given **your own virtual machine** and you’re **root** there – do whatever you want
  - You can run **certain processes**, but you **don’t get root**, call our helpdesk with your configuration requests and we’ll get back to you in 5 business days…

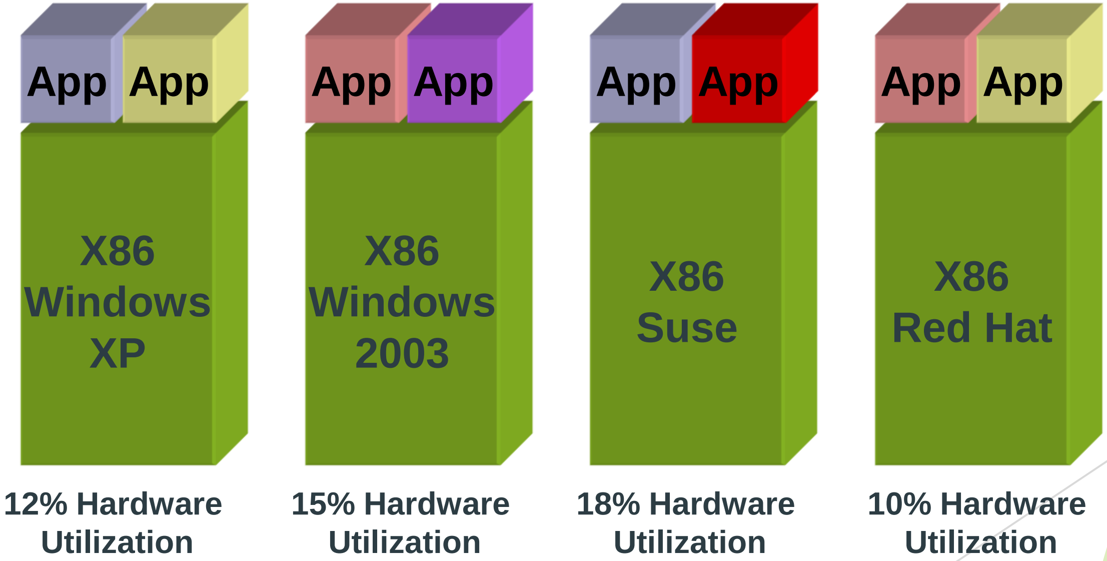
The evolution of virtualization

How did it start?

- Server virtualization has existed for several decades
  - IBM pioneered more than 30 years ago with the capability to “multitask”
- The inception was in specialized, proprietary, high-end server and mainframe systems
- By 1980/90 servers virtualization adoption initiated a reduction
  - Inexpensive x86 hardware platforms
  - Windows/Linux adopted as server OSs

Computing Infrastructure – 2000

- 1 machine → 1 OS → several applications
- Applications can affect each other
- Big disadvantage: machine utilization is very low, most of the times it is below than 25%



Virtualization again…

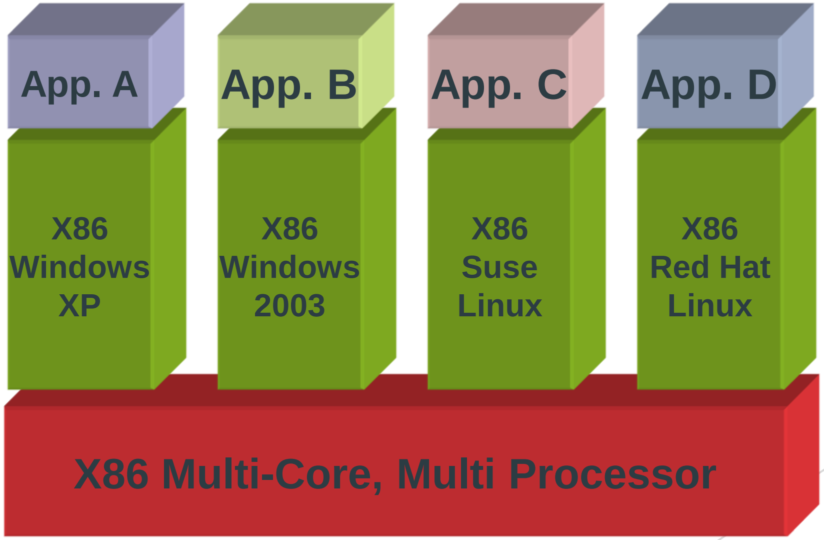
x86 server deployments introduced new IT challenges:

- Low server infrastructure utilization (10-18%)
- Increasing physical infrastructure costs (facilities, power, cooling, etc)
- Increasing IT management costs (configuration, deployment, updates, etc)
- Insufficient failover and disaster protection

The solution for all these problems was to virtualize x86 platforms

Computing Infrastructure - Virtualization

- matches the benefits of high hardware utilization with running several operating systems (applications) in separated virtualized environments
  - Each application It runs in its own operating system
  - Each operating system does not know it is sharing the underlying hardware with others

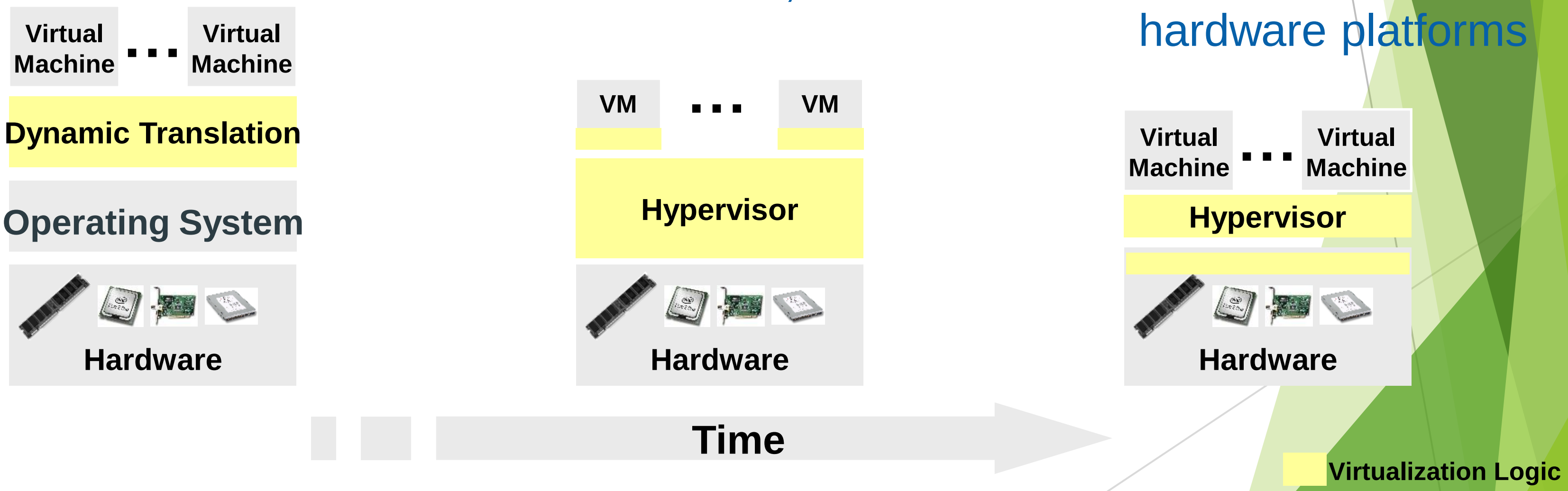


70% Hardware Utilization

Approaches to server virtualization

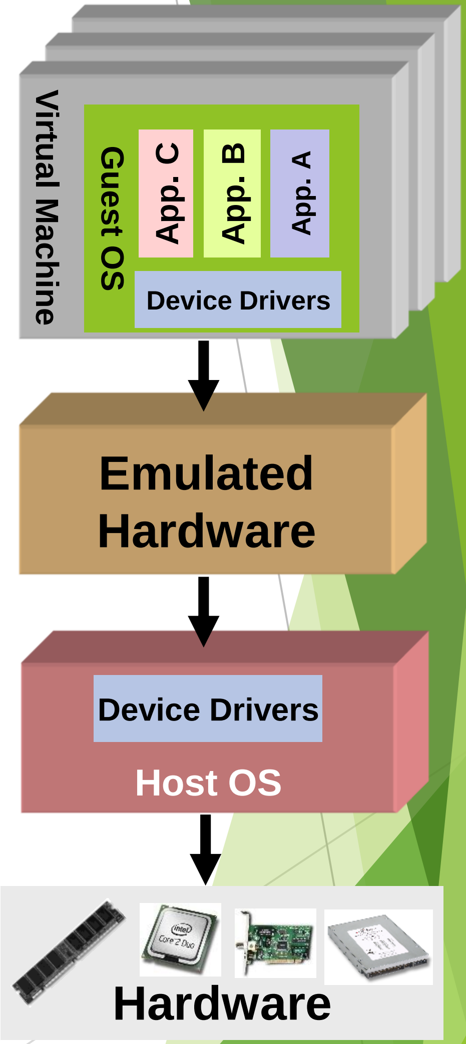
Evolution of Software solutions

- **1st Generation: Full virtualization (Binary rewriting)**
  - Software Based
  - VMware and Microsoft
- **2nd Generation: Paravirtualizatio n**
  - Cooperative virtualization
  - Modified guest
  - VMware, Xen
- **3rd Generation: Silicon-based (Hardware-assisted) virtualization**
  - Unmodified guest
  - VMware and Xen on virtualization-aware hardware platforms



Full Virtualization

- 1st Generation offering of x86/x64 server virtualization
- Dynamic binary translation
  - The emulation layer talks to an operating system which talks to the computer hardware
  - The guest OS doesn't see that it is used in an emulated environment
- All of the hardware is emulated including the CPU
- Two popular open source emulators are QEMU and Bochs

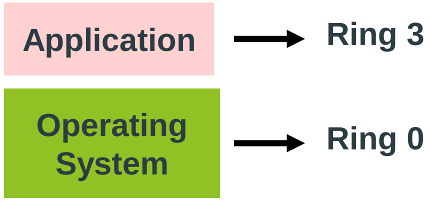


Full Virtualization - Advantages

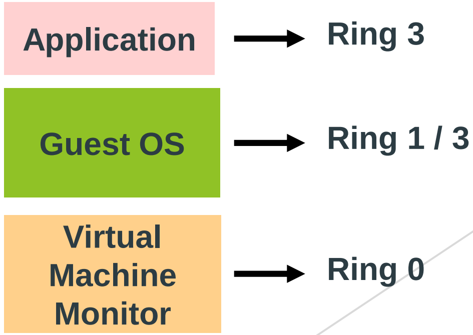
- The emulation layer
  - Isolates VMs from the host OS and from each other
  - Controls individual VM access to system resources, preventing an unstable VM from impacting system performance
- Total VM portability
  - By emulating a consistent set of system hardware, VMs have the ability to transparently move between hosts with dissimilar hardware without any problems
    - It is possible to run an operating system that was developed for another architecture on your own architecture
    - A VM running on a Dell server can be relocated to a Hewlett-Packard server

Full Virtualization - Drawbacks

- Hardware emulation comes with a performance price
- In traditional x86 architectures, OS kernels expect to run privileged code in Ring 0
  - However, because Ring 0 is controlled by the host OS, VMs are forced to execute at Ring 1/3, which requires the VMM to trap and emulate instructions
- Due to these performance limitations, paravirtualization and hardware-assisted virtualization were developed


Application → Ring 3
Operating System → Ring 0

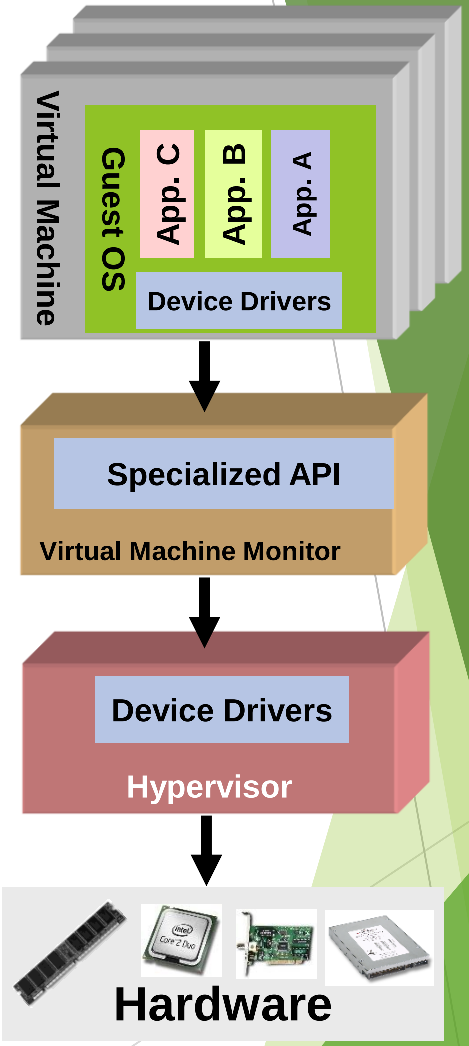
Traditional x86 Architecture


Application → Ring 3
Guest OS → Ring 1 / 3
Virtual Machine Monitor → Ring 0

Full Virtualization

Para-Virtualization

- The Guest OS is modified and thus run kernel-level operations at Ring 1 (or 3)
  - the guest is fully aware of how to process privileged instructions
  - thus, privileged instruction translation by the VMM is no longer necessary
  - The guest operating system uses a specialized API to talk to the VMM and, in this way, execute the privileged instructions
- The VMM is responsible for handling the virtualization requests and putting them to the hardware



Virtual Machine

Guest OS

App. C

App. B

App. A

Device Drivers

Specialized API

Virtual Machine Monitor

Device Drivers

Hypervisor

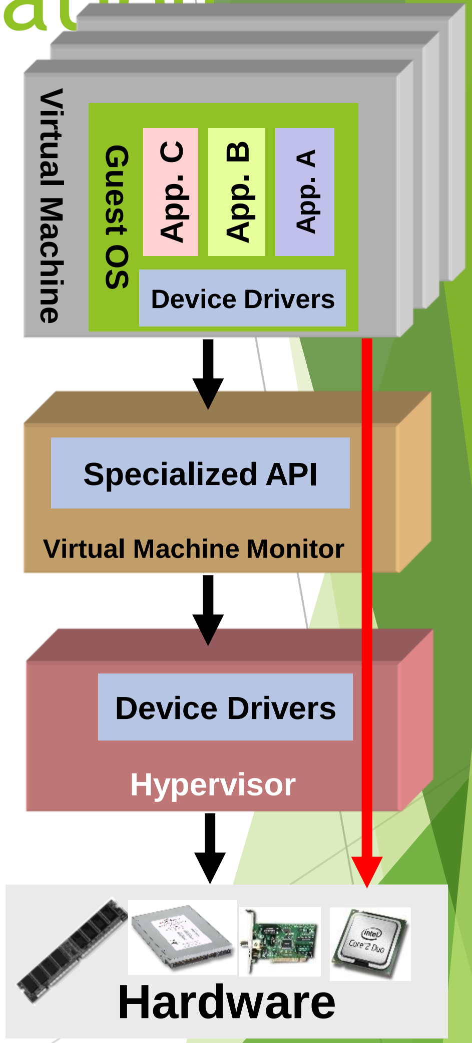
Hardware

Para-Virtualization

- Today, VM guest operating systems are paravirtualized using two different approaches:
  - Recompiling the OS kernel
    - Paravirtualization drivers and APIs must reside in the guest operating system kernel
    - You do need a modified operating system that includes this specific API, requiring a compiling operating systems to be virtualization aware
      - Some vendors (such as Novell) have embraced paravirtualization and have provided paravirtualized OS builds, while other vendors (such as Microsoft) have not
  - Installing paravirtualized drivers
    - In some operating systems it is not possible to use complete paravirtualization, as it requires a specialized version of the operating system
    - To ensure good performance in such environments, paravirtualization can be applied for individual devices
    - For example, the instructions generated by network boards or graphical interface cards can be modified before they leave the virtualized machine by using paravirtualized drivers

Hardware-assisted virtualization

- The guest OS runs at ring 0
- The VMM uses processor extensions (such as Intel®-VT or AMD-V) to intercept and emulate privileged operations in the guest
- Hardware-assisted virtualization removes many of the problems that make writing a VMM a challenge
- The VMM runs in a more privileged ring than 0, a virtual -1 ring is created



Hardware-assisted virtualization

- Pros
  - It allows to run unmodified Oss (so legacy OS can be run without problems)
- Cons
  - Speed and Flexibility
    - An unmodified OS does not know it is running in a virtualized environment and so, it can’t take advantage of any of the virtualization features
      - It can be resolved using paravirtualization partially

Approaches to desktop virtualization

Extending the concept of virtualization for desktops

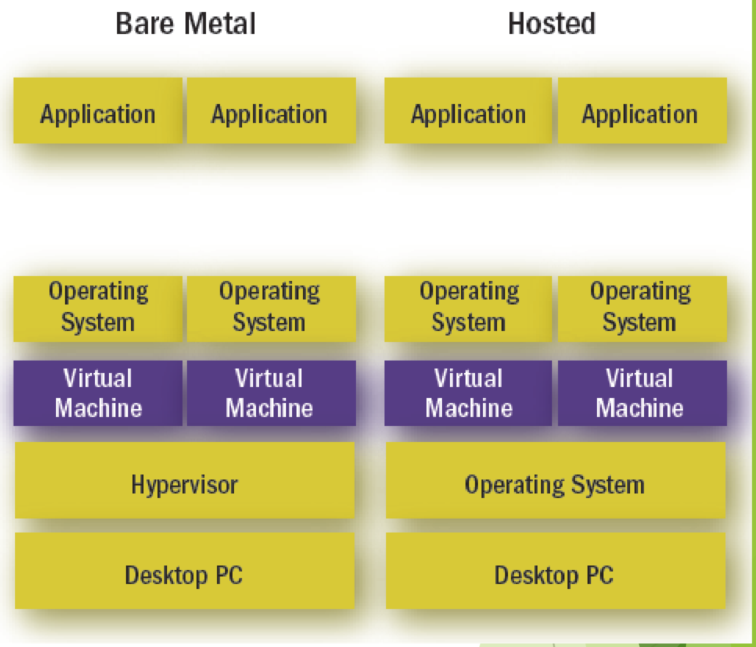
- Servers
  - Hosted virtualization - mainframes
  - VMMs / Bare Metal hypervisors
  - OS virtualization
- Desktops
  - Desktop virtualization
  - Server-side workspace virtualization
  - Client-side workspace virtualization
- Application virtualization
  - Application isolation
  - Application streaming

Desktop Virtualization

- A VMM or hypervisor running on a physical desktop
- Examples include:
  - Microsoft Virtual PC
  - Parallels Desktop for Mac
  - VMware Fusion
  - WINE.
- Use cases include:
  - Emulating Windows games on the Macintosh,
  - Testing code inside VMs
  - Underpinning client-side workspace virtualization

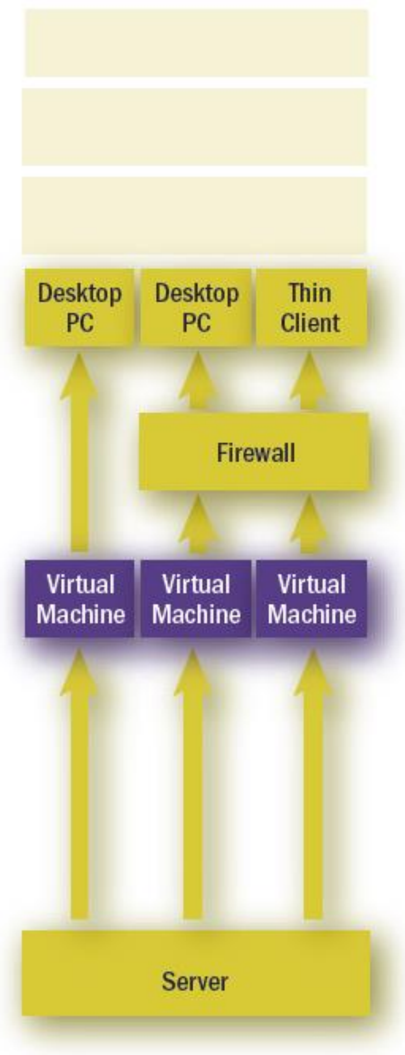
- Desktop hypervisors and VMMs don’t necessarily scale to meet enterprise needs; that’s why most of the providers have server products as well

Desktop virtualization approaches



Server-side workspace virtualization

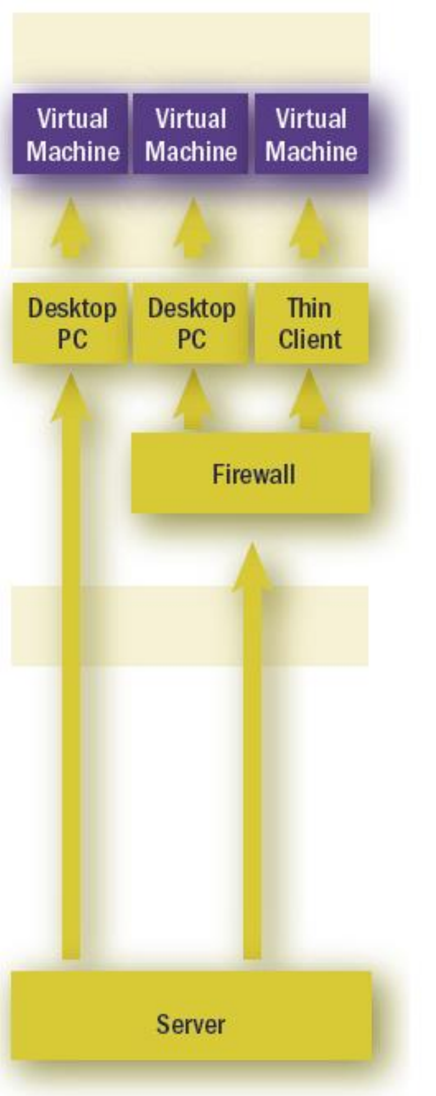
- A workspace (desktop operating system with custom configuration) running inside a virtual machine hosted on a server
- Examples include:
  - VMware VDI
- Use cases include:
  - Centrally managed desktop infrastructure
  - Security enforcement and lockdown



- A pool of virtual workspaces resides on the server. Remote users log into them from any networked device via Microsoft's Remote Desktop Protocol (RDP)
- Users can customize their virtual workspace to their heart's content, while operators enjoy the relatively straightforward task of managing desktop configuration on one central server
- Connection brokers arbitrate between a pool of virtual workspaces residing on a central server
- The biggest problem with server-hosted workspace virtualization is that it's a bandwidth hog. Performance is constrained by the performance of your network

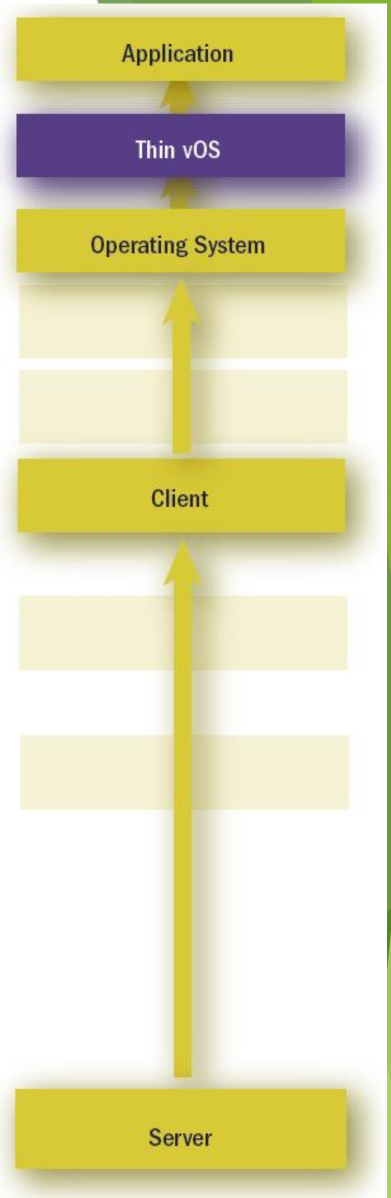
Client-side workspace virtualization

- A workspace (desktop operating system with custom configuration) running inside a virtual machine hosted on a desktop
- Examples include:
  - Kidaro Managed Workspace
  - Sentillion vThere
- Use cases include:
  - Secure remote access
  - Protection of sensitive data for defense, healthcare industries
  - Personal computer running corporate desktops remotely
- A virtual workspace is served out to execute on the client device
- Centralizes management
- Its big advantage over other models is the security and isolation of data and logic on the client
- It’s the right model for organizations that need to ensure the security of environments served to remote users
  - Defense contractors
  - Healthcare providers



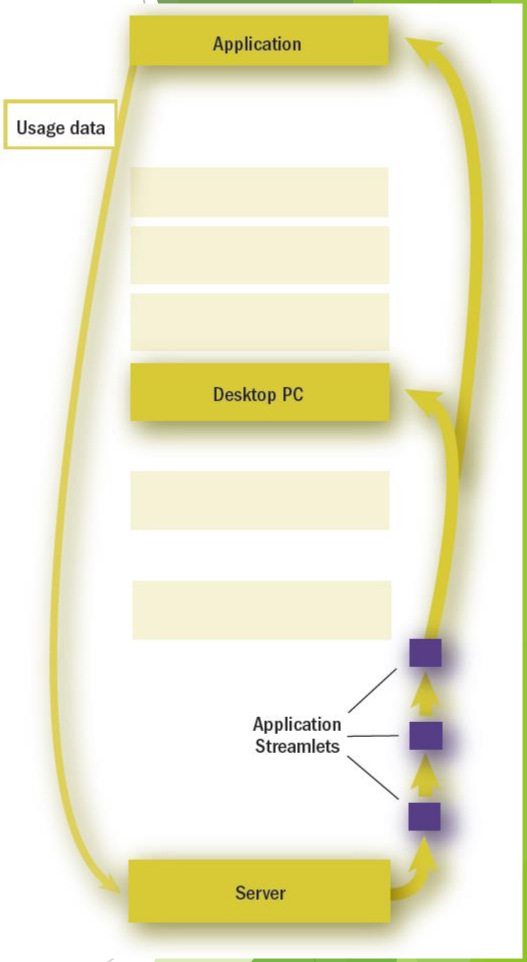
Application Isolation

- An application packaged with its own virtual copies of the operating system resources it might otherwise need to change (registries, file systems, libraries)
- Examples include:
  - Thinstall
  - Trigence
- Use cases include:
  - Preventing DLL hell
  - Sandboxing desktop applications for secure execution
- Applications use a virtual registry (Thinstall) and file system embedded in the package with the application
  - These extra tools insulate applications from changes to and incompatibility with the underlying desktop operating system
- Mostly in Windows, although Linux and Solaris as well
- Drawback: increased footprint of the application package and the correspondingly greater memory requirements



Application Streaming

- Just-in-time delivery of a server-hosted application to the desktop, such that the desktop application can execute before the entire file has been downloaded from the server
- Examples include:
  - AppStream
  - Microsoft SoftGrid
- Use cases include:
  - Managing the number of instances of running applications, in the case of license constraints
- Superset of Application Isolation, including a delivery method and an execution mode
  - You stream the application code to the desktop, where it runs in isolation
- No full PC environment, just the application, so you have to provide a workspace
  - Requires to maintain the client-side operating system and ensuring compatibility. This may be why application streaming, which has been around for a long time (AppStream has already raised over $50m in venture capital), has not really lived up to its early hype.



Questions?