Chapter 12
Cloud Computing

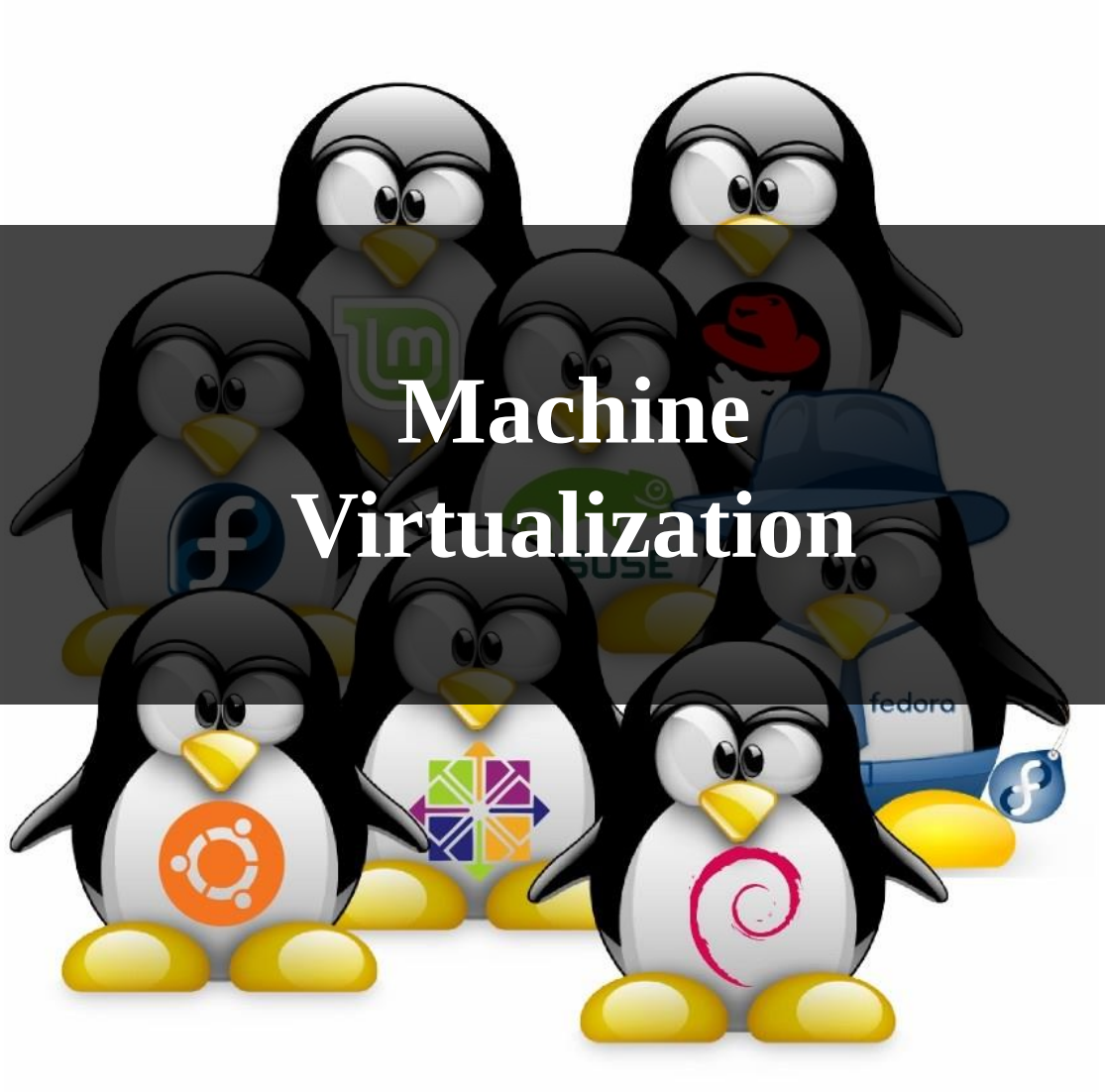
Open Source SW Development
CSE22300

Virtualization

Virtualization Type

- **Memory Virtualization**
  - **Process feels like it has its own address space**
  - **Created by MMU, configured by OS**
- **Storage Virtualization**
  - **Logical View of disks "connected" to a machine**
  - **External pool of storage**
- **CPU Virtualization**
  - **Each process feels like it has its own CPU**
  - **Created by OS preemption and scheduler**

Machine Virtualization



Machine Virtualization

- **Normally all hardware and I/O managed by one operating system**

- **Machine virtualization**
  - **Abstract (virtualize) control of hardware and I/O from the OS**
  - **Partition a physical computer to act like several real machines**
  - **Manipulate memory mappings**
  - **Set system timers**
  - **Access devices**
  - **Migrate an entire OS & its applications from one machine to another**

- **1972: IBM System 370**

Machine Virtualization

- **Privileged vs. unprivileged instructions**
- **Regular applications use unprivileged instructions**
  - **Easy to virtualize**

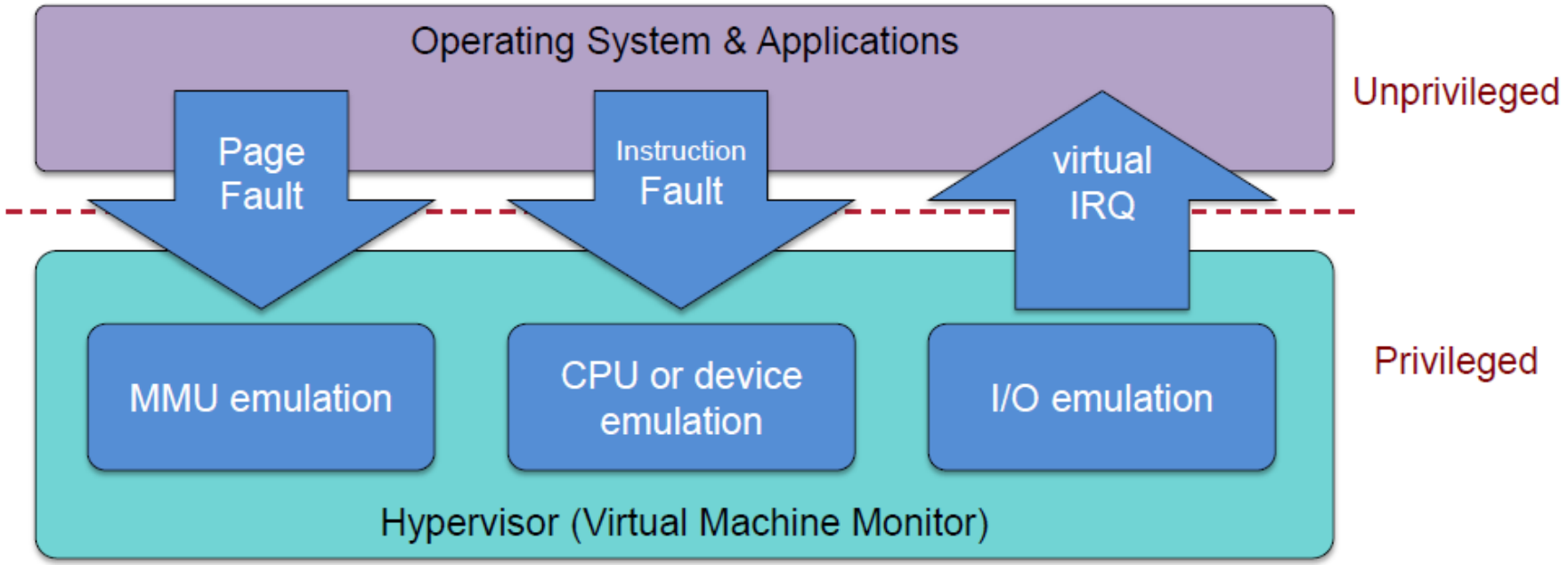
- **If regular applications execute privileged instructions, they trap**
- **VM catches the trap and emulates the instruction**
  - **Trap & Emulate**

Hypervisor

- **Hypervisor: Program in charge of virtualization**
  - **Aka Virtual Machine Monitor**
  - **Provides the illusion that the OS has full access to the hardware**
  - **Arbitrates access to physical resources**
  - **Presents a set of virtual device interfaces to each host**

Hypervisor

- **Application or Guest OS runs until:**
  - **Privileged instruction traps**
  - **System interrupts**
  - **Exceptions (page faults)**
  - **Explicit call: VMCALL (Intel) or VMMCALL (AMD)**



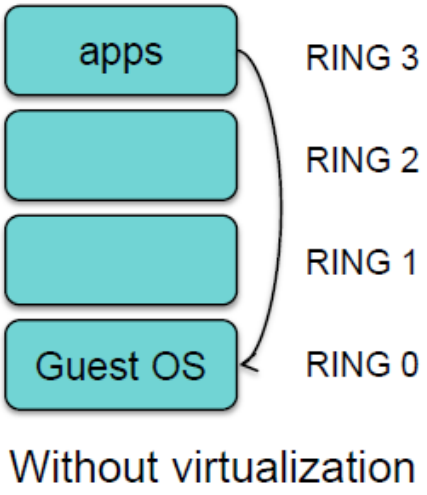
Intel & ARM Didn't Make VM Easy

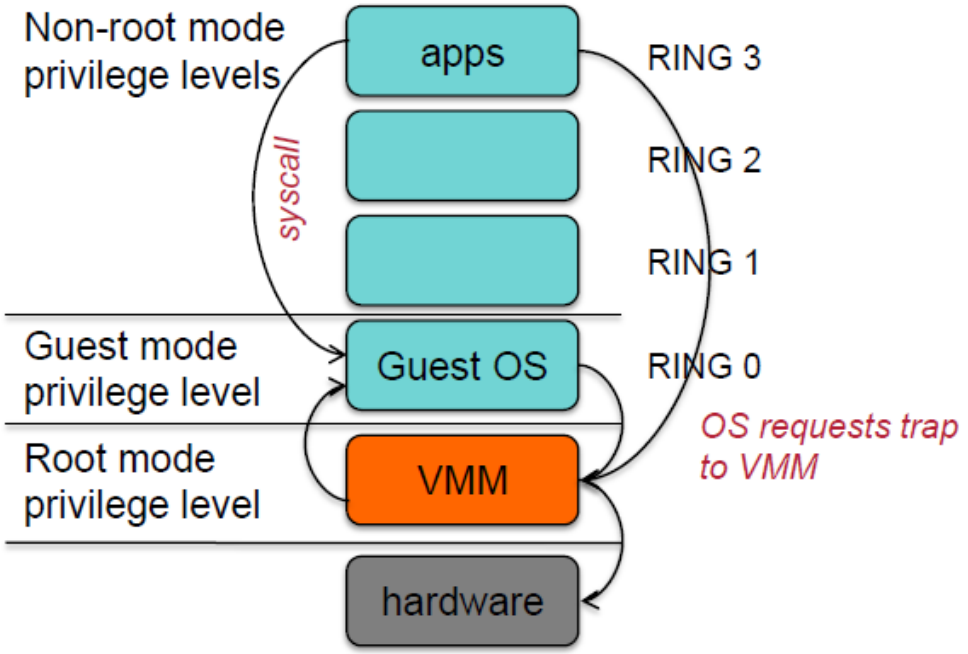
- **Intel/AMD systems prior to Core 2 Duo (2006) did not support trapping privileged instructions**
- **Most ARM architectures also did not trap on certain privileged instructions**
  - **Hardware support added in Cortex-A15 (ARMv7 Virtualization Extension): 2011**

- **Two approaches**
  - **Binary translation (BT)**
    Scan instruction stream on the fly (when page is loaded) and replace privileged instructions with instructions that work with the virtual hardware (VMware approach)
  - **Paravirtualization**
    Don't use non-virtualizable instructions (Xen approach)
    Invoke hypervisor calls explicitly

Hardware support for virtualization

- **Root mode (Intel example)**
  - **Layer of execution more privileged than the kernel**





Architectural Support

- **Intel Virtual Technology**
- **AMD Opteron**

- **Guest mode execution: can run privileged instructions directly**
  - **E.g., a system call does not need to go to the VM**
  - **Certain privileged instructions are intercepted as VM exits to the VMM**
  - **Exceptions, faults, and external interrupts are intercepted as VM exits**
  - **Virtualized exceptions/faults are injected as VM entries**

CPU Architectural Support

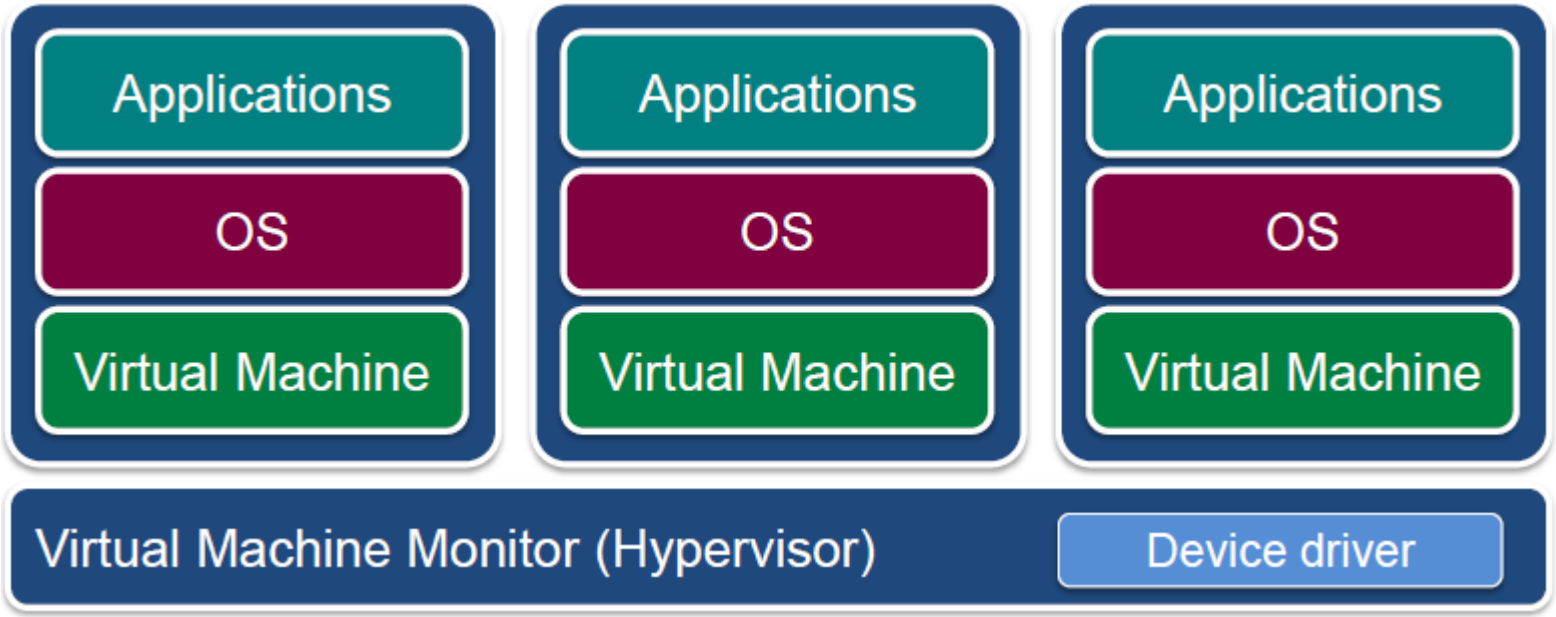
- **Setup**
  - **Turn VM support on/off**
  - **Configure what controls VM exits**
  - **Processor state**
  - **Saved & restored in guest & host areas**
- **VM Entry: go from hypervisor to VM**
  - **Load state from guest area**
- **VM Exit**
  - **VM-exit information contains cause of exit**
  - **Processor state saved in guest area**
  - **Processor state loaded from host area**

Two Approaches to Running VMs

- **Native VM (hypervisor model)**
- **Hosted VM**

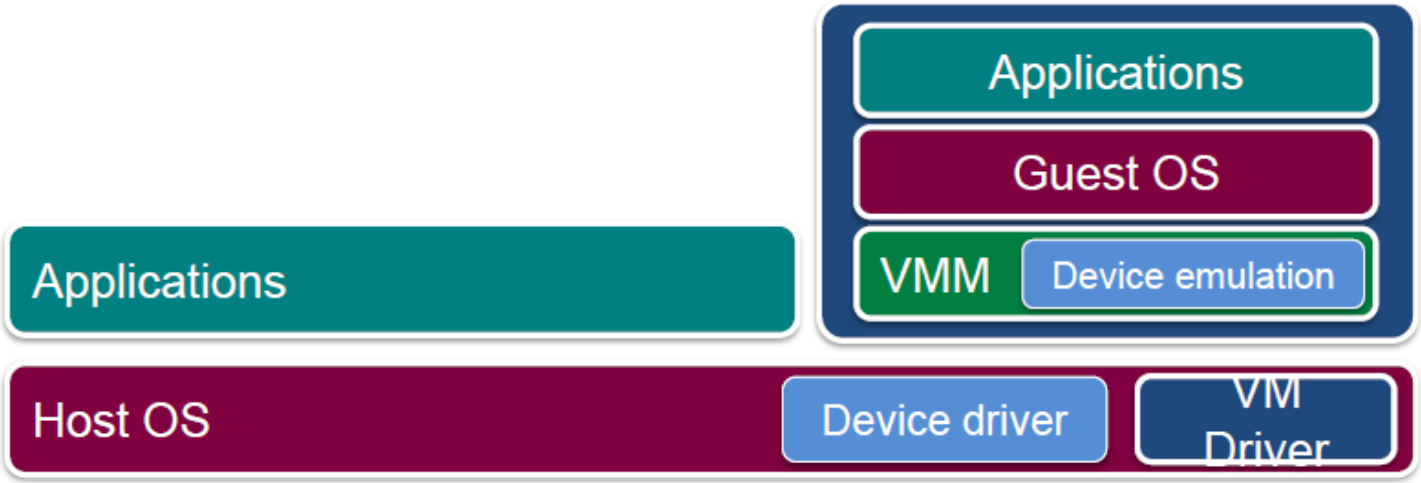
Native Virtual Machine

- **Native VM (or Type 1 or Bare Metal)**
  - **No primary OS**
  - **Hypervisor is in charge of access to the devices and scheduling**
  - **OS runs in "kernel mode" but does not run with full privileges**

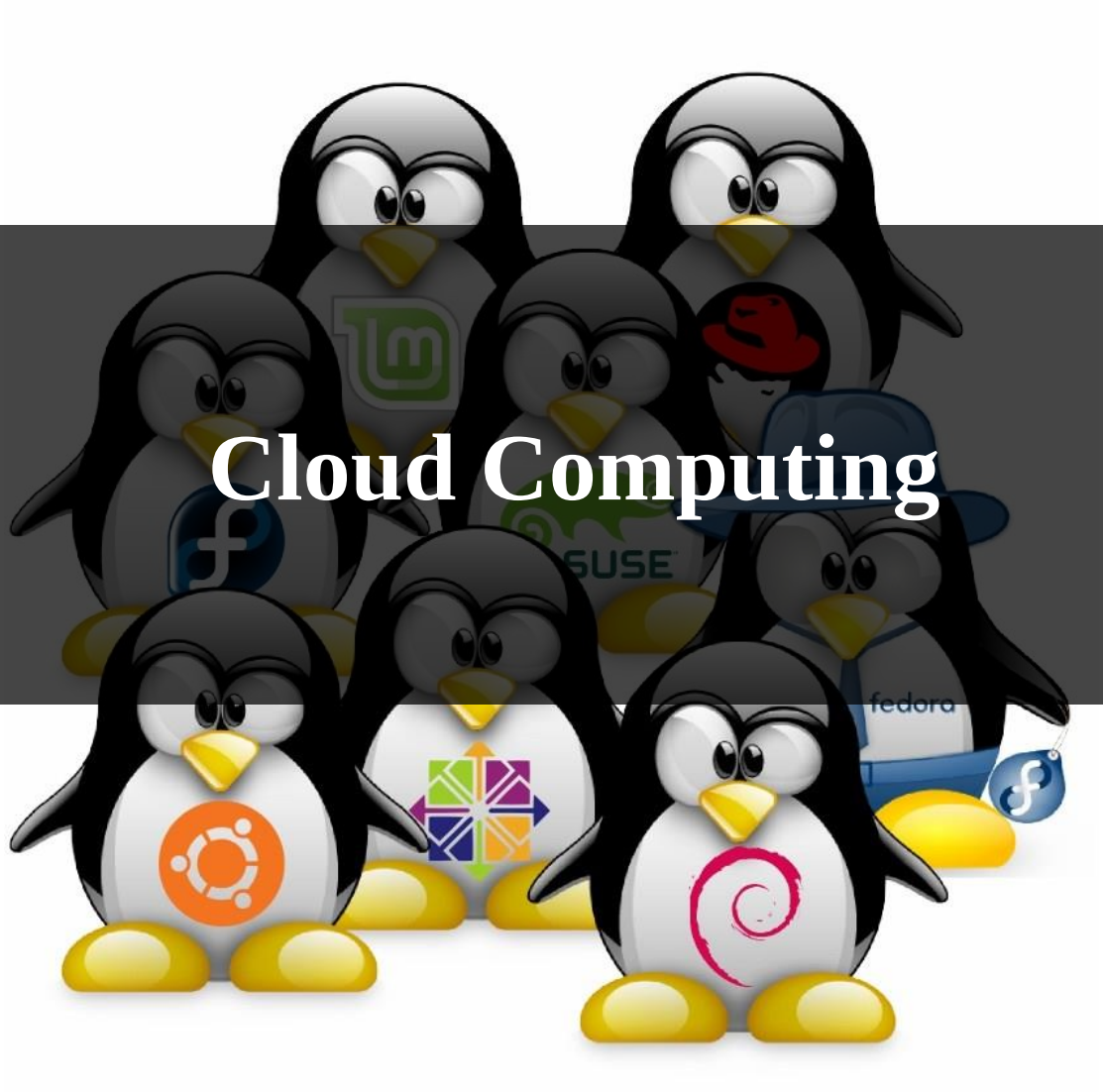


Hosted Virtual Machine

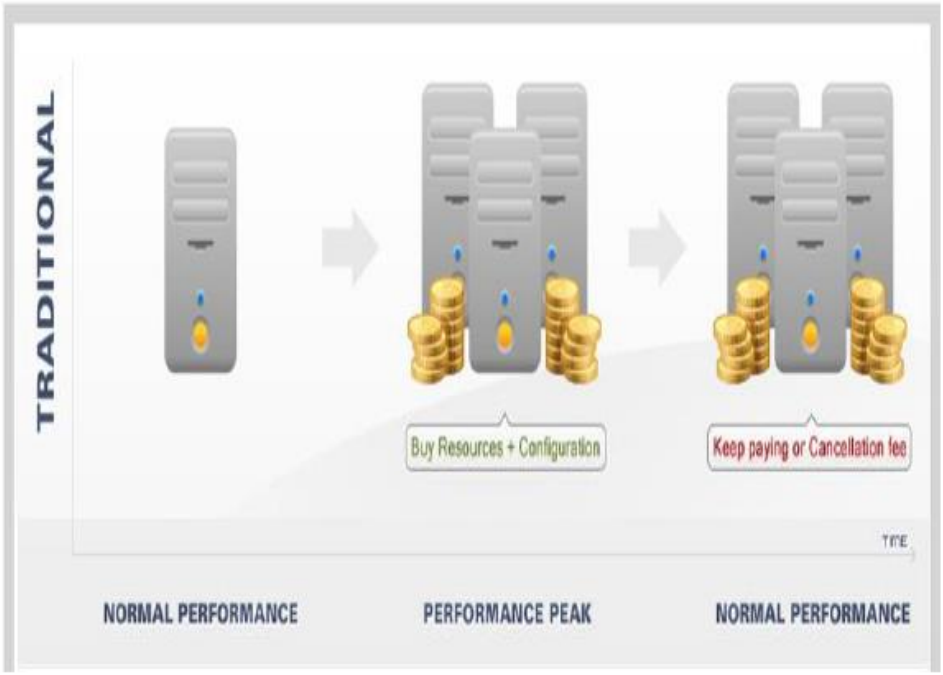
- **Hosted VM**
  - **VMM runs without special privileges**
  - **Primary OS responsible for access to the raw machine**
  - **Lets you use all the drivers available for that primary OS**
  - **Guest operating systems run under a VMM**
  - **VMM invoked by host OS**
  - **Serves as a proxy to the host OS for access to devices**

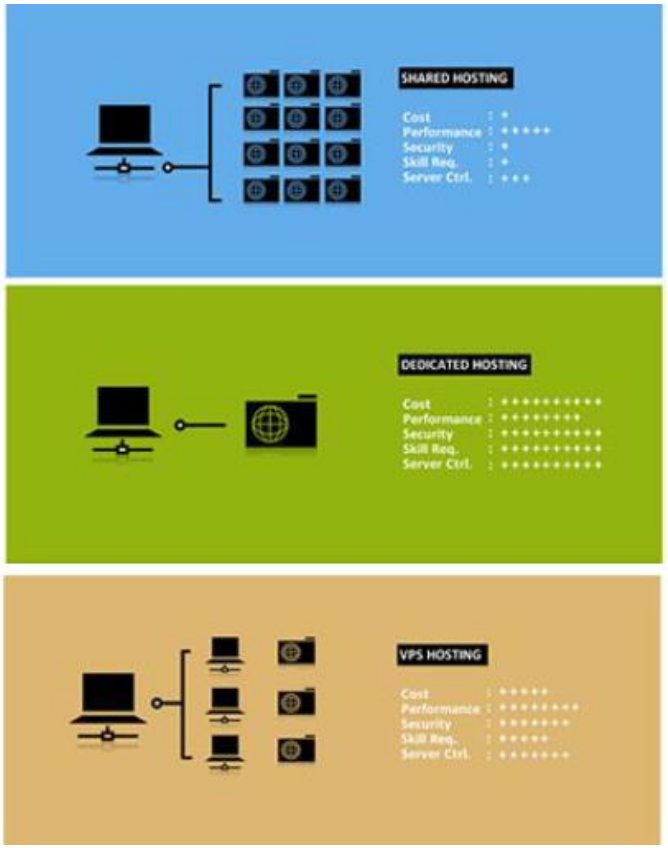


Cloud Computing



Traditional Models





Data Center

- **Collection of servers and computing devices that are networked together and co-located into a single facility**
- **Servers can be configured and set up with appropriate systems and application software**
- **Major online companies have their own data centers, Google, eBay, Amazon**

Cloud Computing

- **Cloud computing is the result of the evolution and adoption of existing technologies and paradigms**

- **Virtualization**
  - **A software that separates a physical computing device into one or more virtual devices**
- **Autonomic computing**
  - **Automation of the process through which a user can provision resources on-demand**
  - **Minimal user involvement, the automated process reduces costs and potential human errors**
- **Service-Oriented Computing**
  - **All resources in cloud computing model are provided as services**
  - **Use of the well-established standards and best practices gained in the domain of SOA to allow global and easy access to cloud services in a standardized way**

What's Cloud Computing?

- **"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." (National Institute of Standards and Technology (NIST), USA).**

Essential Characteristics

- **On-demand self-service**
  - **Computing capabilities**
  - **server and network storage**
  - **provisioned as needed automatically**
- **Broad network access**
  - **Capabilities are available over the network and accessed through standard mechanisms**
- **Resource pooling**
  - **The provider's computing resources are pooled**
  - **Serve multiple consumers using a multi-tenant model**
  - **Different physical and virtual resources dynamically assigned and reassigned according to consumer demand**

Essential Characteristics

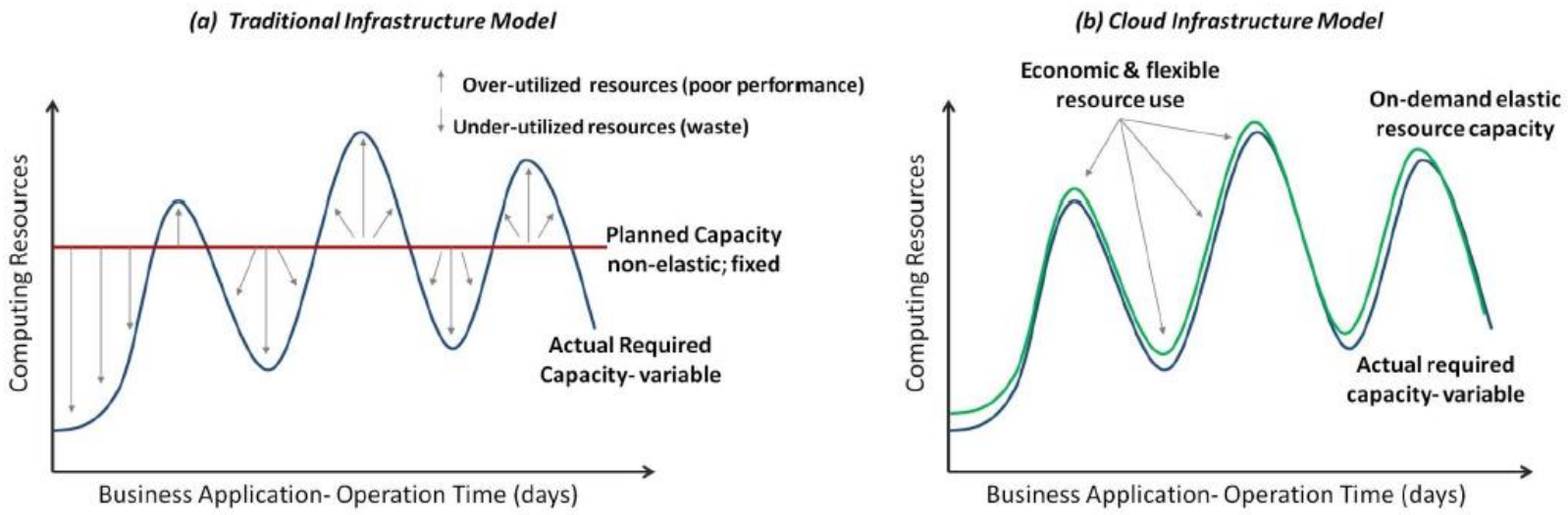
- **Rapid elasticity**
  - **Capabilities can be elastically provisioned and released**
  - **Scale rapidly outward and inward commensurate with demand**
  - **To the consumer, the capabilities available for provisioning in any quantity at any time.**

Case Study

- **Data-Intensive Application**
  - **In 2007, The New York Times decided to make all public domain articles from 1851 - 1922 available free of charge**
  - **11 million articles from 1885 - 1980 - each of which is composed of TIFF images that have to be combined – hugely compute and data-intensive**
  - **Solution - Use Amazon S3 to store the article data (4 TB) and EC2 machines to generate the PDFs which were saved back to S3 from where they are served**

- **Use Hadoop (open-source Map-Reduce implementation) for programming**
- **100 EC2 instances + Hadoop + 24 hours = Job Done!**

Elasticity



- **On-demand computing resources**
  - **e.g., servers, storage**
- **Efficient use of resources**
  - **pay per usage time (pay-as-you-go)**

Elasticity (Auto-Scaling)

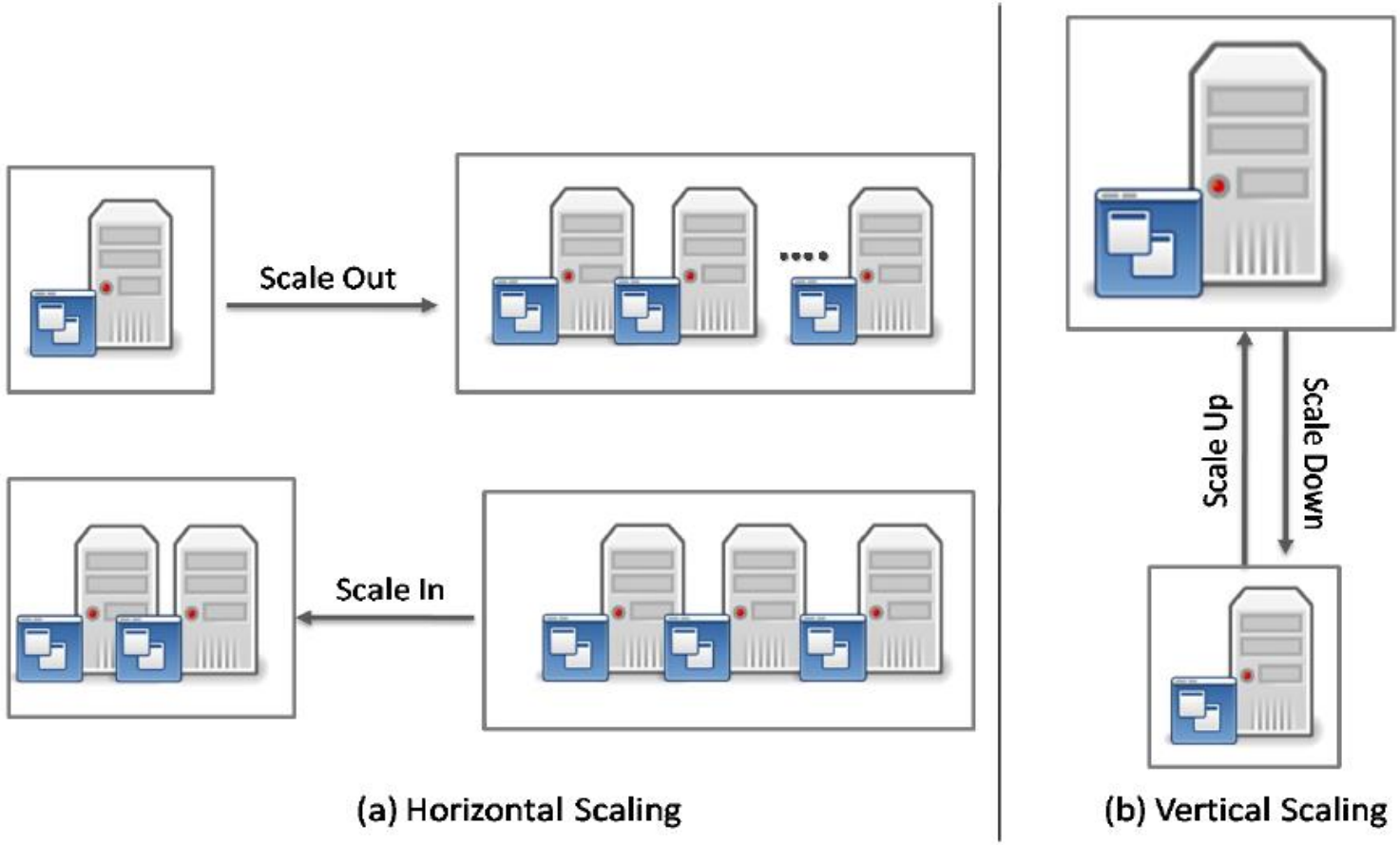
- **Dynamically adapt its computing infrastructure resources in response to variable workload changes over time**
  - **Can be at SaaS, PaaS, IaaS**

- **IaaS Elasticity**
  - **Adding/removing virtual or physical servers**
  - **Increasing/decreasing CPU, memory and storage capacity**
  - **Increasing/decreasing network speed and number of IP addresses**
  - **Increasing/decreasing amount of data transfer and number of data operations/requests of cloud resources**
- **Manual (user interface) vs. automated means (APIs)**
  - **Auto-scaling**

Elasticity (Case Study)

- **Animoto an online video service, makes it easy to make and share videos in just a few minutes**
- **The company launched in 2007 using its own servers, but moved to AWS for additional capacity**
- **When Animoto integrated with Facebook in 2008, attracting 750,000 new users in 3 days, it used AWS to handle the load.**

Types of Scaling



Horizontal vs. Vertical Scaling

- **Horizontal (Scale-out and Scale-in)**
  - **More computing resources (e.g., servers)**
  - **Reliable – fail-over scenario**
  - **Fully automated**
  - **Growing management complexity**

- **Vertical (Scale-up and Scale-down)**
  - **More powerful computing resources – bigger servers**
  - **Single point of failure**
  - **Human intervention**
  - **Reasonable management overhead**

Elasticity (Auto-Scaling) Rules

- **Rule-based mechanism**
  - **Monitor certain resources/application metrics**
  - **Determine when to trigger adding releasing computing resources**
  - **Determine how much computing resources to add/release**
  - **Choose appropriate values for the core thresholds and parameters**

Auto-scaling Rules – Example

Monitor CPU Utilization (CPUUtil) every 1 min. interval

IF CPUUtil > 80% FOR 7 minutes

Add 1 server of small capacity

Wait 5 consecutive 1 min. intervals

IF CPUUtil < 30% FOR 10 minutes

Remove 1 server of small capacity

Wait 7 consecutive 1 min. interval