



# Introduction to Virtualization and Cloud Environments


Saurabh Srivastava

Department of Computer Sc. & Engg.

IIT Kanpur

# At a glance...

- Virtualization
  - Introduction to virtualization
  - Basic Concepts
  - Hypervisors
- Cloud Environments
  - Introduction to cloud
  - Types of cloud environments



# Part – I

## Virtualization

# What we'll cover in this part

- A well known approach towards information gathering is to do so via asking some basic questions about the topic of interest
- We'll get to know the answers to six basic questions about *Virtualization*
  - What, Who, Why, When, Where and How
- We'll then delve into more technical details of virtualization
- We'll take a look at different types of *hypervisors* – the tools that enable virtualization
- Most of the content presented here is taken from the book titled [Virtualization Essentials](#) by Matthew Portnoy

# What is virtualization?

- “Virtualization in computing often refers to the abstraction of some physical component into a logical object”
- **Virtual Machines, Virtual Disks, Virtual Networks** etc. are some examples of what can be virtualized
- Virtual versions of physical entities are expressed generally as **one or more files**
- For example, a Virtual Machine can be described using a file that describe its configuration (such as CPU, memory, storage, network etc.) and some other files that represent the virtual disks attached to the machine
- Modifying a virtual resource often implies modifying one or more corresponding files

# Who does virtualization commercially?

- The idea and execution of virtualization is not new
- IBM mainframes had implementations of the idea in early 1970s
- The first commercially available solution to provide virtualization for x86 computers came from VMware in 2001 (namely, **ESX 1.0** and **GSX 1.0**)
- A parallel open-source offering called **Xen** arrived two years later
- In addition, currently, there are a number of virtualization offerings available, provided by different vendors including VMWare, IBM, Microsoft, Oracle etc.
- Some examples of platform specific solutions are **Solaris Zones**, **BSD jails**, **PowerVM** on IBM machines etc.
- **VMWare Workstation** and **Oracle Virtualbox** are examples of virtualization offerings that can run on wider range of underlying platforms

# Why should we care for virtualization? <sup>(1/2)</sup>

- Moore's law is an observation on expansion of processing power over a period of time
  - Simply stated, it says that the processing power **roughly doubles every 18 months**
  - Moore's law applies not just to processing power but to many other related technologies, including Memory Capacities too
- Organizations generally replace their servers in three to five years time because they may no longer be enough to fit their IT requirements
  - This happens because of rapidly growing databases and the applications built to process data in them efficiently within acceptable timing constraints
- Some organizations prefer leasing over purchasing, yet the overall constraints still apply to them as well

# Why should we care for virtualization? (2/2)

- This is where virtualization can be a saviour
- Recall that unlike their physical counterparts, virtual resources are generally “a set of files”
- Modifying or upgrading a virtual resource is almost always much easier and quicker than doing so with a physical resource
- For example, updating the Memory in a Virtual Machine would mean changing the value of a field in a configuration file
- Organizations thus, are moving preferring to deploy their applications on *Virtual Servers* rather than buying Physical Servers to host them



# When should we go for virtualization?

- Virtualization has its own drawbacks too
  - Since virtualization often involves a layer of abstraction between the application and the hardware, it can lead to performance degradation
- However, if the applications running on a physical server do not make full use of the hardware resources, consolidation can lead to **better hardware utilisation**
  - Condensing multiple servers on to one (multiple VMs on single Physical Host) is called **Consolidation** and the number of servers condensed is called **Consolidation Ratio** (e.g. for 8 VMs running on a physical host, consolidation ratio is 8:1)
  - The consolidation ratios of the first generation of x86 hypervisors were in the range of 5:1
- Even a modest consolidation ratio of 4:1 could remove three-quarters of the servers in a Datacentre !

# Where can we do virtualization?

- Virtualization is not just an option for Datacentres
- There are virtualization solutions which you can run on your Desktops or Laptops
  - We'll see a demo of creating Virtual Machines on a standalone laptop shortly
- Virtualization provides the underlying foundation to build *Cloud Environments*
  - We'll see how virtualization is used in the Cloud in the second part of the presentation
- Virtualization can also be limited to specific aspects such as “virtualizing only the desktop” and not the whole system
  - **Citrix's XenDesktop** and **VMWare's View** are to popular solutions for Desktop Virtualization
- There are also solutions available to support “application virtualization” such as **Microsoft's App-V** and **VMware's ThinApp**

# How can we use virtualization?

- We'll have a look at one particular virtualization solution that can run on your laptops
- Oracle's Virtualbox is a free and open-source virtualization solution available for x86 architecture computers running Windows, Linux, OS X, Solaris, FreeBSD etc.
- We will now have a live demo of how we can use Virtualbox to create some Virtual Machines on our laptops

# Virtualization Basics <sup>(1/2)</sup>

- The first paper to talk formally about virtualization came in 1973 by Popek and Goldbers titled “*Formal Requirements for Virtualizable Third Generation Architectures*”
- They defined the concept of a Virtual Machine Monitor (**VMM**), which in today’s terminology is called a **hypervisor**
- According to the paper, a VMM needs to exhibit three properties in order to correctly satisfy their definition
  - **Fidelity:** The environment it creates for the VM is essentially identical to the original (hardware) physical machine
  - **Isolation or Safety:** The VMM must have complete control of the system resources
  - **Performance:** There should be little or no difference in performance between the VM and a physical equivalent

# Virtualization Basics <sup>(2/2)</sup>

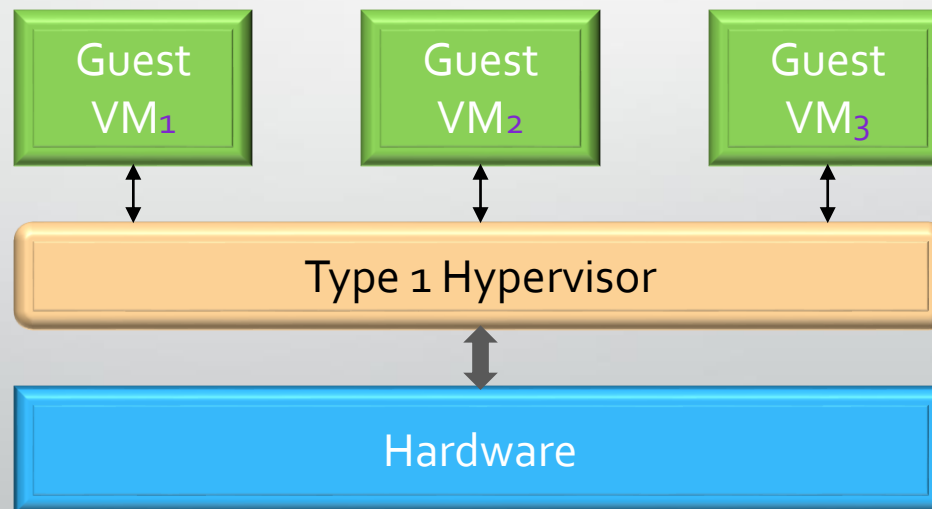
- The **hypervisor** is a layer of software that resides below the virtual machines and above the hardware
- The hypervisor manages the interactions between each virtual machine and the hardware that the guests all share

# Hypervisors <sup>(1/4)</sup>

- It is possible to build hypervisor capabilities either directly above the hardware, or interface with an Operating System managing the hardware
- **Type 1** hypervisors (also referred to as a **bare-metal** implementation) run directly on the server hardware without an operating system beneath it
  - Type 1 hypervisors can only support guest operating systems with a compatible kernel
- A **Type 2** hypervisor (also known as a **hosted hypervisor** implementation) itself is an application that runs atop a traditional operating system, and intermediates interaction between the guest and the host operating systems
  - Type 2 hypervisors can support a wide range of guest operating systems since there is a software layer that sits between the guest and the host

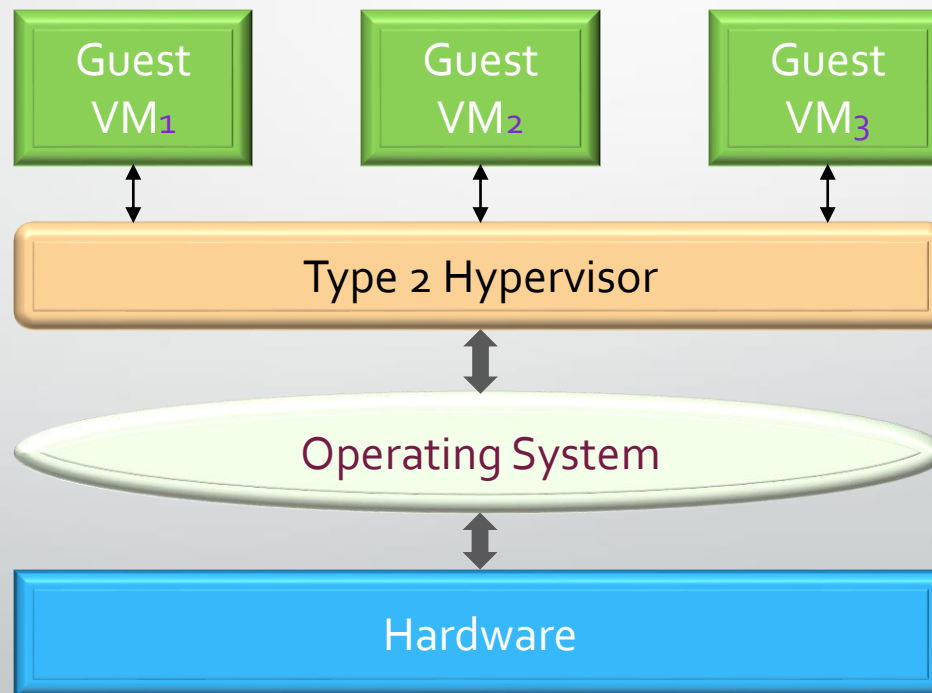
# Hypervisors (2/4)

- A Type 1 hypervisor can directly communicate with the hardware resources in the stack below it, making it much more efficient than the Type 2 hypervisor
- Type 1 hypervisors are also considered to be more secure than Type 2 hypervisors
  - This is because the guest operations are *passed through* to the hardware, and, as such, a mischievous guest cannot affect the hypervisor on which it is supported



# Hypervisors (3/4)

- Type 2 hypervisors are usually easy to install and deploy because much of the hardware configuration work, such as networking and storage, has already been covered by the operating system





# Hypervisors (4/4)

Type 1 Hypervisors	Type 2 Hypervisors
Runs over the host hardware directly	Runs over the host operating system
Efficient since there is no abstraction layer	Slower because of translations from guest to host
The guests must have the same OS base	The guests can run any OS base
Problems in one guest generally remains isolated	Problems in one guest can affect hypervisor process, hence affecting other guests too
Examples: <b>VMware ESX</b> , <b>Microsoft Hyper-V</b> and many <b>Xen</b> variants	Examples: <b>VMware Workstation</b> , <b>Microsoft Virtual Server</b> and <b>Oracle Virtualbox</b>



# Part – II

## Cloud Environments

# What we'll cover in this part

- We'll cover in short how the concept of cloud computing evolved due to business needs
- We'll see two different taxonomies of cloud environment, based on the service model and deployment arrangements
- Most of the content presented here is taken from [\*Architecting Software for the Cloud\*](#) online course by Prof. T. V. Prabhakar and Prof. Balwinder Sodhi

# The idea of “a cloud”

- Cloud computing is not a recent concept
  - The idea of time sharing systems in the 1960s can be considered as the initial point
- The term “cloud” itself depicts a model to support **distributed computing**
- The Datacentres generally are grossly underused
- Businesses wanted a way out to tap the unused resources
- Remember, virtualization can help here
  - Loads distributed over several, underused physical servers can be consolidated over virtual servers running on the same physical server
  - The overall resource utilization goes up, and excluding peculiar situations where peak load windows of the VMs coincide or overlap, the model works fairly well

# Definitions <sup>(1/2)</sup>

- The National Institute of Standards and Technology (NIST), United States, define five characteristics of a cloud offering
- The document can be accessed publicly [here](#)
- **Broad Network Access**
  - The services deployed on a cloud are available over a network, and are accessed via the broad range of Network Protocols such as HTTP, HTTPS, SSH etc.
- **Resource Pooling**
  - On a Cloud Platform, resources such as storage units and computation units are pooled by the Cloud Provider, which can then be used to serve multiple Cloud applications at the same time

# Definitions <sup>(2/2)</sup>

- **On-demand Self-Service**
  - Consumers can provision computing capabilities without human interaction
- **Measured Service**
  - Usage of resources can be monitored, controlled, and reported
  - Provides transparency for both the provider and consumer
- **Rapid elasticity**
  - Computing capabilities can be rapidly and elastically provisioned to quickly scale up and rapidly released to scale down
  - To the consumer, the capabilities available for provisioning often appear to be unlimited

# Taxonomy of cloud environments

- Based on their deployment and the offered services, cloud environments can be classified into different categories
- Classification on the basis of offered services:
  - **Infrastructure as a Service**
  - **Platform as a Service**
  - **Software as a Service**
- Classification on the basis of deployment:
  - **Public Cloud**
  - **Private Cloud**
  - **Hybrid Cloud**

# Infrastructure as a Service

- **IaaS** provides *fundamental computing resources* as the offered service for usage
- This includes resources like **computation, storage, networking** etc.
- In simpler terms, an IaaS provider would provide offerings at a core level, such as a **Virtual Machine**, a **Storage Array** and a (per-project) **Firewalled Network**
- There is little or no restrictions on how the virtual resources are used, managed, updated or communicate with each other
- The user is solely responsible for even basic tasks such as installing an operating system on the VMs, defining networks and subnets, and applying patches or updates to any deployed applications
- Provides maximum control to the user, but requires maximum efforts to maintain



# Platform as a Service

- **PaaS** offerings provide support for programming environments, development or deployment stacks, APIs etc.
- For example, the user is offered a **LAMP** stack as a service, or an environment to deploy **Java applications**, say via JAR or WAR files
- The details of underlying infrastructure, like the configuration of the machine on which LAMP is installed, remains opaque from the user
- The provider is responsible to maintain and upgrade packages related to offered services, such as MySQL Server or the JDK
- The user gets the liberty to make application specific configurations, such as ability to do URL rewriting in apache, or add custom JARs to the CLASSPATH
- Takes away the headache of maintaining the stack from user at the cost of full control

# Software as a Service

- **SaaS** offerings provide an already ready to use service or product, with possible user-specific customizations
- Typically no programming is involved and/or allowed in SaaS offerings
- A website with the option to choose custom themes, banners, logos etc. could be considered a SaaS example
- The user has no control or knowledge of the underlying infrastructure
- The user can make tweaks, usually via a GUI, to his/her own preferences, but cannot alter any fundamental behaviour of the application
- Usually suitable for those cases where user needs are fairly standard, and a little customization makes the services acceptable to a wide range of customers
- Provides very little control, hence minimal manual intervention is needed to maintain

# IaaS, PaaS and SaaS

IaaS	PaaS	SaaS
Product or Service	Product or Service	Product or Service
Runtimes, APIs, Middleware	Runtimes, APIs, Middleware	Runtimes, APIs, Middleware
Operating System	Operating System	Operating System
Virtualization	Virtualization	Virtualization
Hardware	Hardware	Hardware
<b>Example:</b> AWS EC2, AWS S3	<b>Example:</b> Google App Engine, IBM Bluemix	<b>Example:</b> Google Sites, Google Mail

You Manage

Vendor Manages

# Public Cloud

- In a public cloud environment, the cloud services are **open to the public** for subscription
- Services can be obtained and put to use easily and quickly
- Public cloud providers maintain their own datacentres, virtualize resources, and then offer them commercially to be bought and used by customers
- Alleviates businesses of the need to maintain their own IT infrastructure
- The *pay-as-you-go* model allows businesses to start off without heavy initial investments and hardware procurement headache
- **Amazon web Services** (AWS) is a public cloud provider which provides a variety of services including IaaS and PaaS offerings for commercial usage
- Some other public cloud vendors are **Microsoft Azure, Google** and **Rackspace**

# Private Cloud

- Private Clouds are created and maintained by an organisation or an institution for its own private use
- The organisation holds the control over the applications, data and information flow within the cloud
- Typically deployed and operated within the premises of the organisation
- Requires installation of tools or frameworks which can manage virtualization as well as aggregation and management of virtualized resources
- **Openstack** is a commonly used open-source solution towards building private clouds
- Other similar solutions include **OpenNebula**, **Eucalyptus** as well as solutions offered by vendors such as **Nutanix**

# Hybrid Cloud

- A composition of two or more clouds – say a public cloud and a private cloud
- Member cloud entities are bound together, yet they remain separate entities
- In the most general setting, in-house infrastructure of an organisation is augmented or supplemented with resources taken from a public cloud
- Tries to take best of both worlds – sensitive data and applications can run on the private cloud, while some applications can *burst* to public cloud as and when internal resources reach their limits
  - Cloud bursting refers to a scenario where an application resides in a private cloud during the normal operation, but takes up resources on-demand in case of a surge in load
- General use-cases include computation offshoring for specific tasks, such as running Big Data analytics on a public cloud, while running enterprise applications locally



# That's it !!

If you are not in a hurry to go out, any Questions?