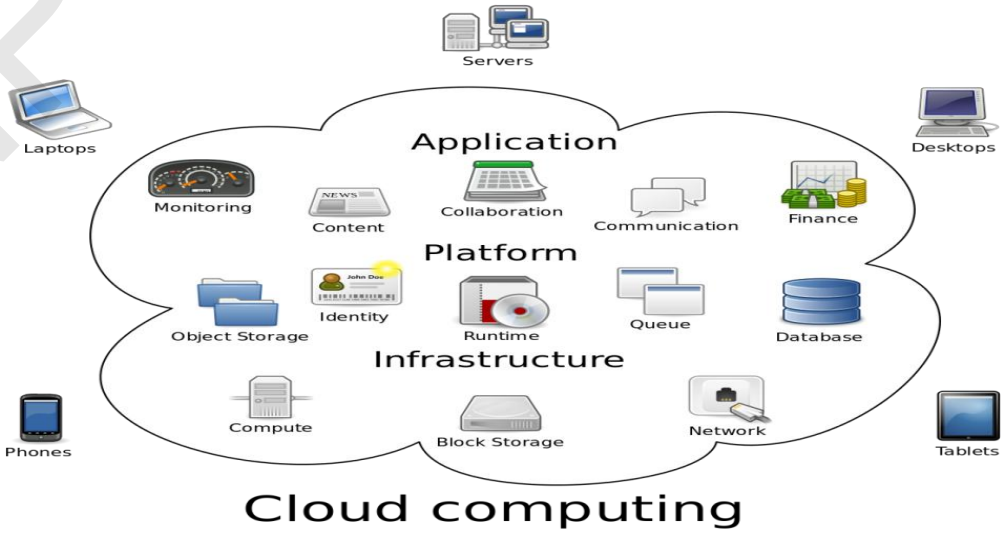
Introduction to Cloud Computing

Dr. Rajiv Misra
Associate Professor
Dept. of Computer Science & Engg.
Indian Institute of Technology Patna
rajivm@iitp.ac.in

Preface

Content of this Lecture:

- In this lecture, we will discuss a brief introduction to Cloud Computing and also focus on the aspects *i.e*. Why Clouds, What is a Cloud, Whats new in todays Clouds and also distinguish Cloud Computing from the previous generation of distributed systems.



Scalable Computing Over the Internet

- Evolutionary changes that have occurred in **distributed and cloud computing** over the past 30 years, **driven by applications with variable workloads and large data sets .**
- Evolutionary changes in machine architecture, operating system platform, network connectivity, and application workload.
- **Distributed computing** system uses multiple computers to solve large-scale problems over the Internet. Thus, **distributed computing becomes data-intensive and network-centric.**
- **The emergence of computing clouds** instead demands high-throughput computing (HTC) systems built with distributed computing technologies.
- **High-throughput computing (HTC)** appearing as computer clusters, service-oriented architecture, computational grids, peer-to-peer networks, Internet clouds, and the future Internet of Things.

The Hype of Cloud: Forecasting

- Gartner in 2009 – Cloud computing revenue will soar faster than expected and will **exceed $150 billion** by 2013. It will represent 19% of IT spending by 2015.
- IDC in 2009: "Spending on IT cloud services will triple in the next 5 years, reaching **$42 billion**."
- Forrester in 2010 – Cloud computing will go from **$40.7 billion** in 2010 to **$241 billion** in 2020.
- Companies and even federal/state governments using cloud computing now: **fbo.gov**

Many Cloud Providers

- AWS: Amazon Web Services
  - EC2: Elastic Compute Cloud
  - S3: Simple Storage Service
  - EBS: Elastic Block Storage
- Microsoft Azure
- Google Compute Engine/AppEngine
- Rightscale, Salesforce, EMC, Gigaspaces, 10gen, Datastax, Oracle, VMWare, Yahoo, Cloudera
- And 100s more…

service cloud

Categories of Clouds

- Can be either a (i) public cloud, or (ii) private cloud
- **Private clouds** are accessible only to company employees
- **Public clouds** provide service to any paying customer:
  - **Amazon S3 (Simple Storage Service):** store arbitrary datasets, pay per GB-month stored
  - **Amazon EC2 (Elastic Compute Cloud):** upload and run arbitrary OS images, pay per CPU hour used
  - **Google App Engine/Compute Engine:** develop applications within their App Engine framework, upload data that will be imported into their format, and run

Customers Save: Time and Money

- "With AWS, a new server can be up and running in **three minutes** compared to **seven and a half weeks** to deploy a server internally and a **64-node Linux cluster** can be online in five minutes (compared with three months internally."
- "With Online Services, reduce the IT **operational costs** by roughly **30%** of spending"
- "A private cloud of virtual servers inside its datacenter has saved nearly **crores of rupees annually**, because the company can share computing power and storage resources across servers."
- 100s of startups can harness large computing resources without buying their own machines.

What is a Cloud?

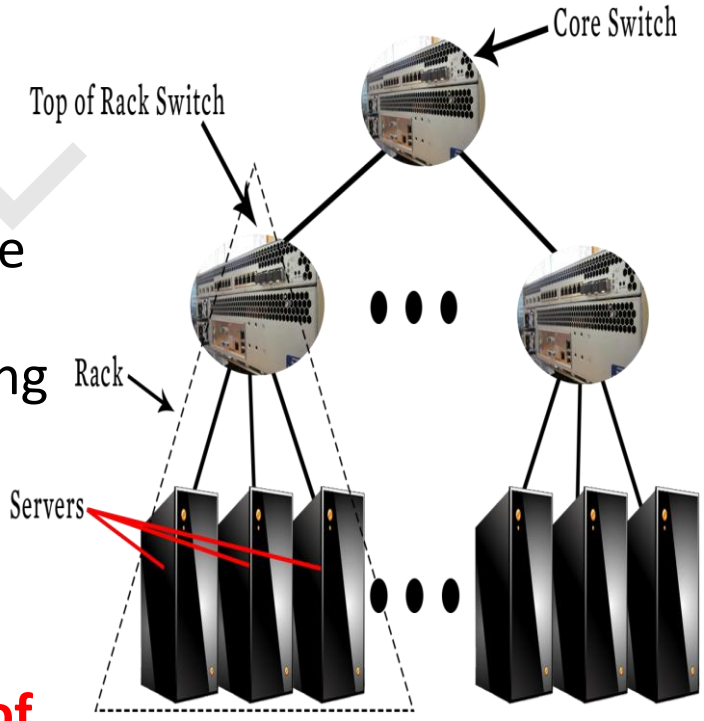
- Advances in virtualization make it possible to see the growth of Internet clouds **as a new computing paradigm**.
- i.e. dramatic differences between developing software for millions to use **as a service** versus distributing software to run on their PCs."

History:

- **In 1984, John Gage** Sun Microsystems gave the slogan, **"The network is the computer."**
- **In 2008, David Patterson** UC Berkeley said, **"The data center is the computer."**
- Recently, **Rajkumar Buyya** of Melbourne University simply said: **"The cloud is the computer."**
- Some people view **clouds as grids** or **clusters** with changes through virtualization, since clouds are anticipated to process huge data sets generated by the traditional Internet, social networks, and the future IoT.

What is a Cloud?

- **A single-site cloud (as known as "Datacenter") consists of**
  - Compute nodes (grouped into racks)
  - Switches, connecting the racks
  - A network topology, e.g., hierarchical
  - Storage (backend) nodes connected to the network
  - Front-end for submitting jobs and receiving client requests
  - (Often called "three-tier architecture")
  - Software Services

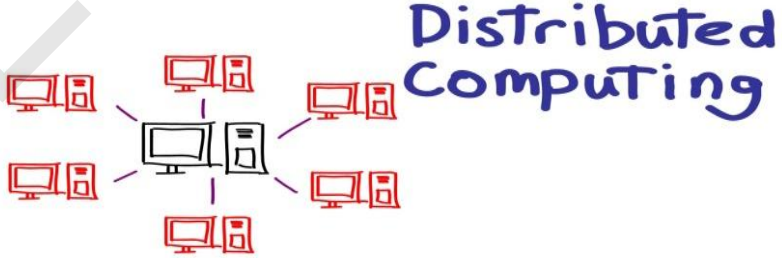


- **A geographically distributed cloud consists of**
  - Multiple such sites
  - Each site perhaps with a different structure and services

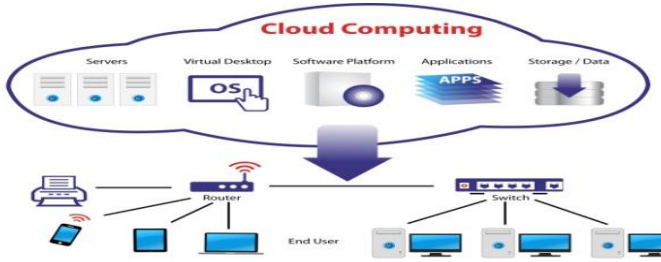
Computing Paradigm Distinctions

- Cloud computing overlaps with distributed computing.

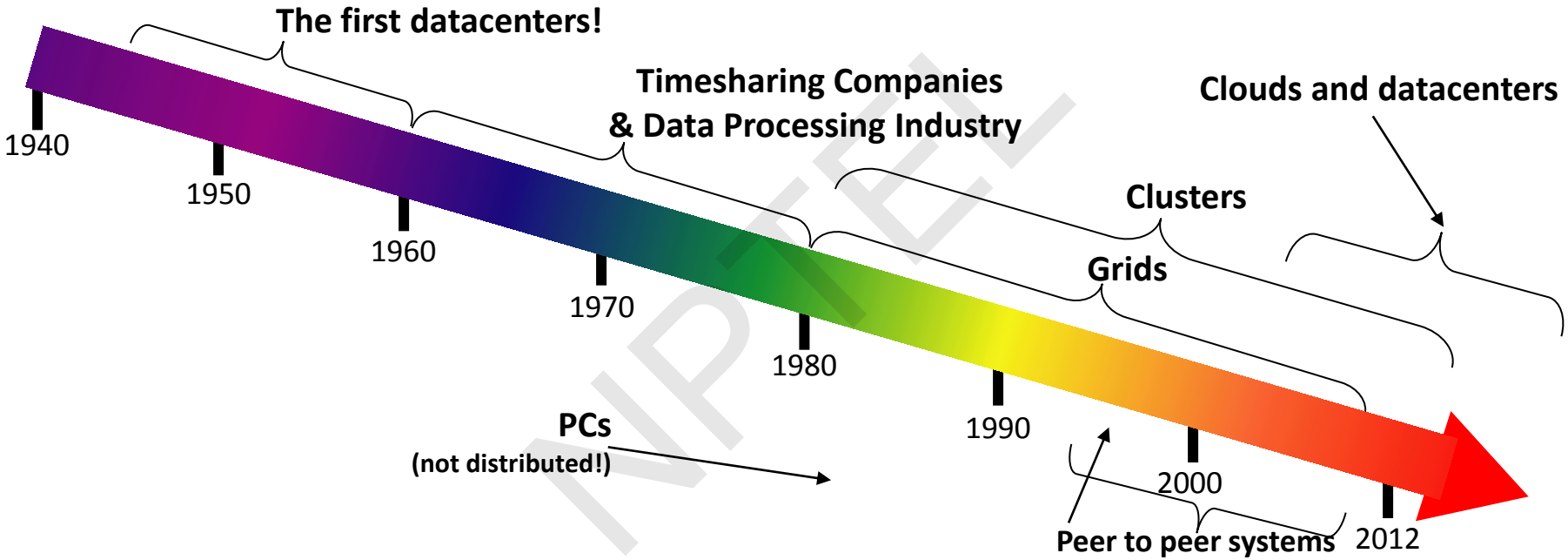
- **Distributed computing:** A ***distributed system*** consists of multiple autonomous computers, having its own memory, communicating through ***message passing***.



- **Cloud computing:** Clouds can be built with physical or virtualized resources over large data centers that are distributed systems. Cloud computing is also considered to be a form of ***utility computing or service computing.***



"A Cloudy History of Time"



"A Cloudy History of Time"

First large datacenters: ENIAC, ORDVAC, ILLIAC
Many used vacuum tubes and mechanical relays

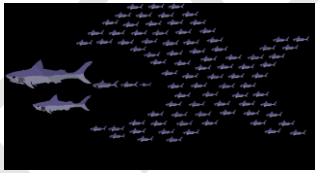
1940

1950

1960

IBM
xerox
Honeywell

1970

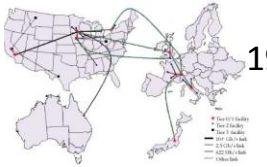


Berkeley NOW Project
Supercomputers
Server Farms (e.g., Oceano)

CRAY
THE SUPERCOMPUTER COMPANY

1980

Open Science Grid



1990

2000

P2P Systems (90s-00s)
- Many Millions of users
- Many GB per day

LimeWire
KaZaA
BitTorrent

2012

Clouds

Data Processing Industry
- 1968: $70 M. 1978: $3.15 Billion

Timesharing Industry (1975):
- Market Share: Honeywell 34%, IBM 15%,
- Xerox 10%, CDC 10%, DEC 10%, UNIVAC 10%
- Honeywell 6000 & 635, IBM 370/168,
  Xerox 940 & Sigma 9, DEC PDP-10, UNIVAC 1108

Grids (1980s-2000s):
- GriPhyN (1970s-80s)
- Open Science Grid and Lambda Rail (2000s)
- Globus & other standards (1990s-2000s)

Scalable Computing Trends: Technology

- **Doubling Periods** – storage: 12 months, bandwidth: 9 months, and CPU compute capacity: 18 months (what law is this?)
- **Moore's law** indicates that processor speed doubles every 18 months.
- **Gilder's law** indicates that network bandwidth has doubled each year in the past.

- Then and Now
  - Bandwidth
    - 1985: mostly 56Kbps links nationwide
    - 2015: Tbps links widespread
  - Disk capacity
    - Today's PCs have TBs, far more than a 1990 supercomputer

The Trend toward Utility Computing

- Aiming towards autonomic operations that can be self-organized to support dynamic discovery. Major computing paradigms are composable with ***QoS and SLAs (service-level agreements).***
- In 1965, MIT's Fernando Corbató of the Multics operating system envisioned a computer facility operating “like a power company or water company”.
- **Plug** your thin client into the computing Utility **and Play** Intensive Compute & Communicate Application
- **Utility computing** focuses on a business model in which customers receive computing resources from a paid service provider.
- All **grid/cloud platforms are regarded as utility service providers**.

Features of Today's Clouds

I. **Massive scale:** Very large data centers, contain tens of thousands sometimes hundreds of thousands of servers and you can run your computation across as many servers as you want and as many servers as your application will scale.

II. **On-demand access:** Pay-as-you-go, no upfront commitment.

- And anyone can access it

III. **Data-intensive Nature:** What was MBs has now become TBs, PBs and XBs.

- Daily logs, forensics, Web data, etc.

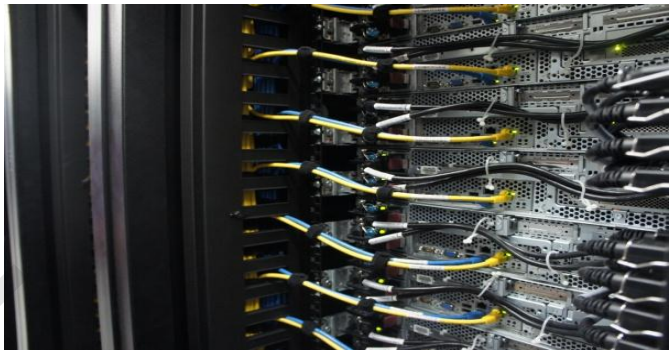
IV. **New Cloud Programming Paradigms:** MapReduce/Hadoop, NoSQL/Cassandra/MongoDB and many others.

- **Combination of one or more of these gives rise to novel and unsolved distributed computing problems in cloud computing.**

I. Massive Scale

- **Facebook [GigaOm, 2012]**
  - 30K in 2009 -> 60K in 2010 -> 180K in 2012
- **Microsoft [NYTimes, 2008]**
  - 150K machines
  - Growth rate of 10K per month
  - 80K total running Bing
  - In 2013, Microsoft Cosmos had 110K machines (4 sites)
- **Yahoo! [2009]:**
  - 100K
  - Split into clusters of 4000
- **AWS EC2 [Randy Bias, 2009]**
  - 40K machines
  - 8 cores/machine
- **eBay [2012]: 50K machines**
- **HP [2012]: 380K in 180 DCs**
- **Google: A lot**

What does a datacenter look like from inside?



Lots of Servers

Power and Energy

- WUE = Annual Water Usage / IT Equipment Energy (L/kWh)
  - low is good
- PUE = Total facility Power / IT Equipment Power
  - low is good (e.g., Google~1.11)

Off-site

On-site

Cooling

- Air sucked in
- Combined with purified water
- Moves cool air through system

II. On-demand access: *AAS Classification

- **On-demand:** renting vs. buying one. E.g.:
  - AWS Elastic Compute Cloud (EC2): a few cents to a few $ per CPU hour
  - AWS Simple Storage Service (S3): a few cents per GB-month
- **HaaS: Hardware as a Service**
  - Get access to barebones hardware machines, do whatever you want with them, Ex: Your own cluster
  - Not always a good idea because of security risks
- **IaaS: Infrastructure as a Service**
  - Get access to flexible computing and storage infrastructure. **Virtualization** is one way of achieving this. subsume HaaS.
  - Ex: Amazon Web Services (AWS: EC2 and S3), OpenStack, Eucalyptus, Rightscale, Microsoft Azure, Google Cloud.

II. On-demand access: *AAS Classification

- **PaaS: Platform as a Service**
  - Get access to flexible computing and storage infrastructure, coupled with a software platform (often tightly coupled)
  - Ex: Google's AppEngine (Python, Java, Go)

- **SaaS: Software as a Service**
  - Get access to software services, when you need them. subsume SOA (Service Oriented Architectures).
  - Ex: Google docs, MS Office on demand

III. Data-intensive Computing

- **Computation-Intensive Computing**
  - Example areas: MPI-based, High-performance computing, Grids
  - Typically run on supercomputers (e.g., NCSA Blue Waters)
- **Data-Intensive**
  - Typically store data at datacenters
  - Use compute nodes nearby
  - Compute nodes run computation services

- In data-intensive computing, the **focus shifts from computation to the data**:
- CPU utilization no longer the most important resource metric, instead I/O is (disk and/or network)

IV. New Cloud Programming Paradigms

- Easy to write and run highly parallel programs in new cloud programming paradigms:
  - **Google:** MapReduce and Sawzall
  - **Amazon:** Elastic MapReduce service (pay-as-you-go)
  - Google (MapReduce)
    - Indexing: a chain of 24 MapReduce jobs
    - ~200K jobs processing 50PB/month (in 2006)
  - **Yahoo!** (Hadoop + Pig)
    - WebMap: a chain of several MapReduce jobs
    - 300 TB of data, 10K cores, many tens of hours (~2008)
  - **Facebook** (Hadoop + Hive)
    - ~300TB total, adding 2TB/day (in 2008)
    - 3K jobs processing 55TB/day
  - NoSQL: MySQL is an industry standard, but Cassandra is 2400 times faster

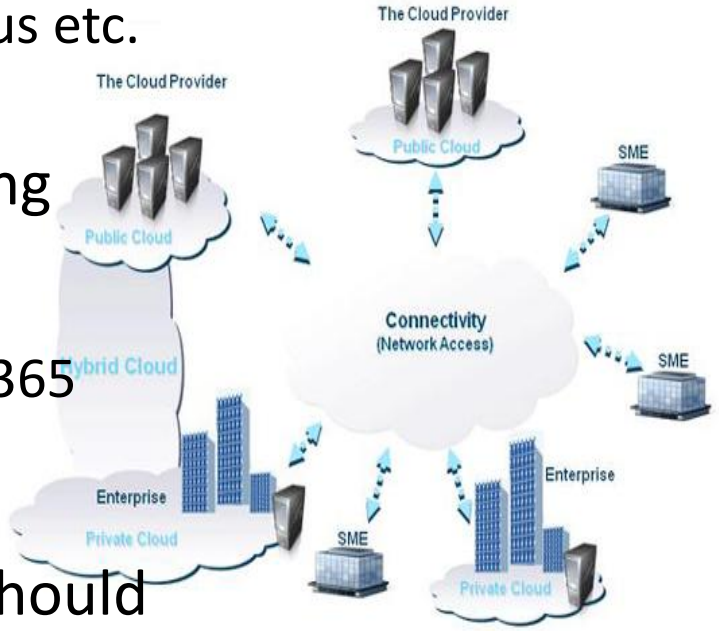
Two Categories of Clouds

- Can be either a (i) public cloud, or (ii) private cloud

- **Private clouds** are accessible only to company employees
- Example of popular vendors for creating private clouds are VMware, Microsoft Azure, Eucalyptus etc.

- **Public clouds** provide service to any paying customer
- Examples of large public cloud services include Amazon EC2, Google AppEngine, Gmail, Office365 and Dropbox etc.

- You're starting a new service/company: should you use a public cloud or purchase your own private cloud?



Single site Cloud: to Outsource or Own?

- Medium-sized organization: wishes to run a service for *M* months
  - Service requires 128 servers (1024 cores) and 524 TB
- **Outsource** (e.g., via AWS): *monthly* cost
  - S3 costs: $0.12 per GB month. EC2 costs: $0.10 per CPU hour (costs from 2009) Storage = $ 0.12 X 524 X 1000 ~ $62 K
  - Total = Storage + CPUs = $62 K + $0.10 X 1024 X 24 X 30 ~ $136 K
- **Own:** monthly cost
  - Storage ~ $349 K / *M* Total ~ $ 1555 K / *M* + 7.5 K (includes 1 sysadmin / 100 nodes)
    - using 0.45:0.4:0.15 split for hardware:power: network and 3 year lifetime of hardware

Breakeven analysis: **more preferable to own if:**

- $349 K / *M* < $62 K (storage)
- $ 1555 K / *M* + 7.5 K < $136 K (overall)

Breakeven points

- *M* > 5.55 months (storage)
- *M* > 12 months (overall)

-Startups use clouds a lot
-Cloud providers benefit monetarily most from storage

Conclusion

- Clouds build on many previous generations of distributed systems

- Characteristics of cloud computing problem

  - **Scale, On-demand access, data-intensive, new programming**

Virtualization

Dr. Rajiv Misra

Associate Professor

Dept. of Computer Science & Engg.

Indian Institute of Technology Patna

rajivm@iitp.ac.in

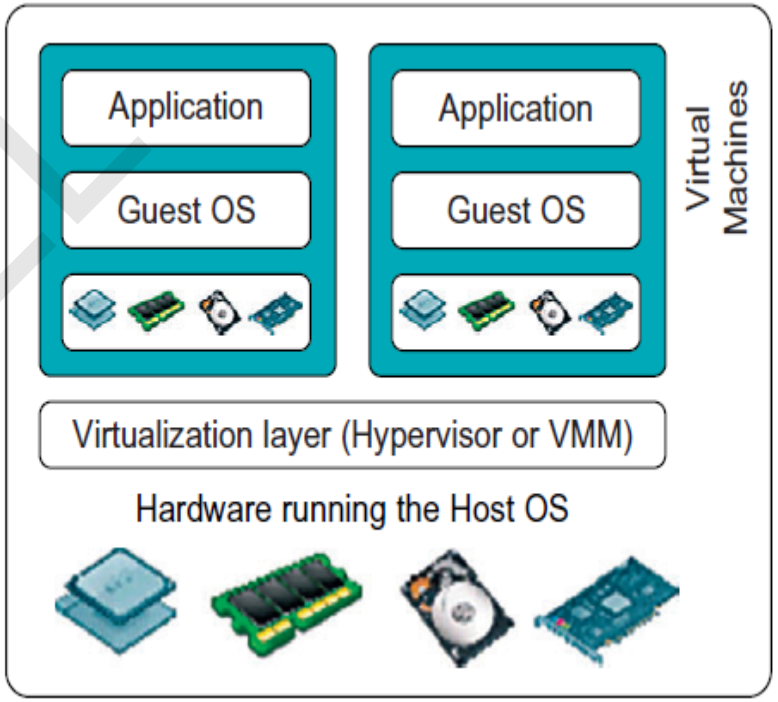
Preface

Content of this Lecture:

- In this lecture, we will discuss virtualization technology its importance, benefits, different models and key approaches to virtualization in CPU, Memory and Device Virtualization.

What is Virtualization ?

- Virtualization is originated in the 1960s at IBM.
- Virtualization **allows** concurrent execution of multiple OSs (and their applications) on the same physical machine.
- Virtual resources = **each OS thinks that it "owns" hardware resources**
- Virtual machine (VM) = **OS+ applications + virtual resources** (**guest domain**)
- Virtualization layer = **management of physical hardware** (**virtual machine monitor, hypervisor**)



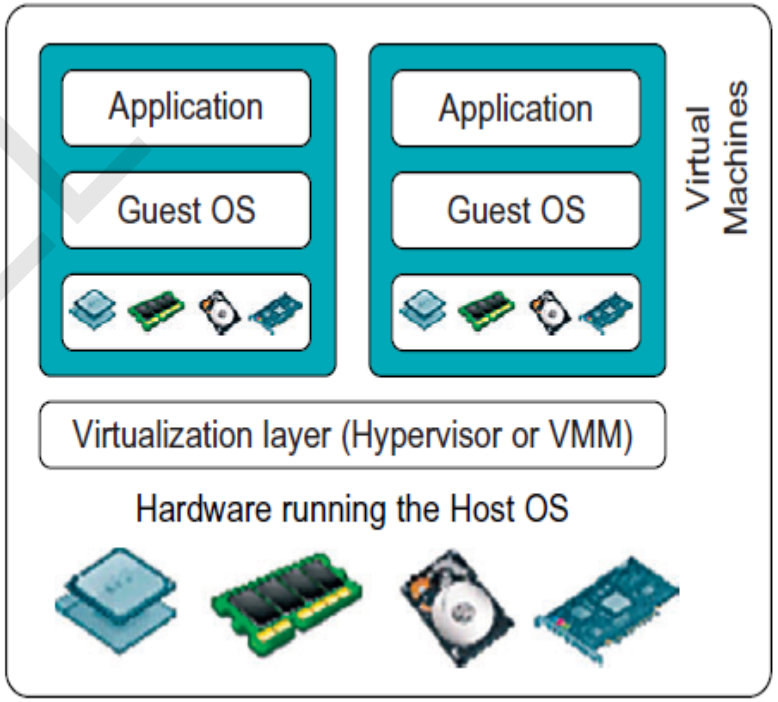
Defining Virtualization

- A virtual machine is an **efficient, isolated duplicate of the real machine.**

- Supported by a **virtual machine monitor (VMM):**

  1. Provides environment **essentially identical with the original machine**
  2. Programs show **at worst only minor decrease in speed.**
  3. VMM is in **complete control of system resources.**

  (This means that the virtual machine monitor has full control to make decisions, who accesses which resources and when)



VMM Goals: Fidelity
Performance
Safety & isolation

Benefits of Virtualization

- **Consolidation:** It is this ability to run multiple virtual machines, with their operating systems and applications on a single physical platform.

  - Decrease cost, improve manageability (with fewer admins and with fewer electrical bills)

- **Migration:** Migrate the OS in the applications from one physical machine to another physical machine.

  - Greater availability of the services, improve reliability

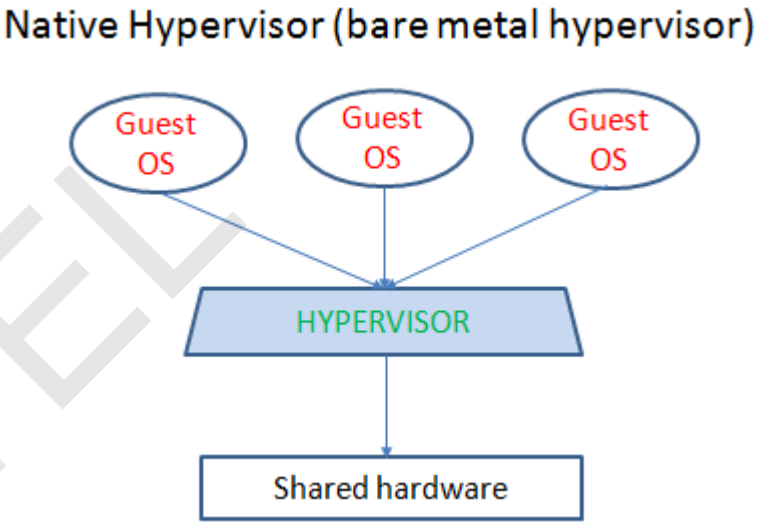
- **Security:** As the OS and the applications are nicely encapsulated in a virtual machine. It becomes more easy to contain any kinds of bugs, or any kinds of malicious behavior, to those resources that are available to the virtual machine only, and not to potentially affect the entire hardware system.

- **Some other benefits:** Debugging, Provide affordable Support for legacy OSs

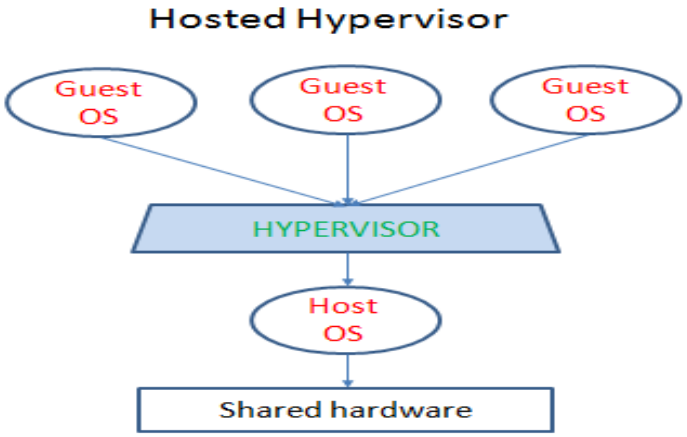
Virtualization Models

The two popular models for virtualization are called:

1. Bare-metal hypervisor
or Native Hypervisor (Type 1)



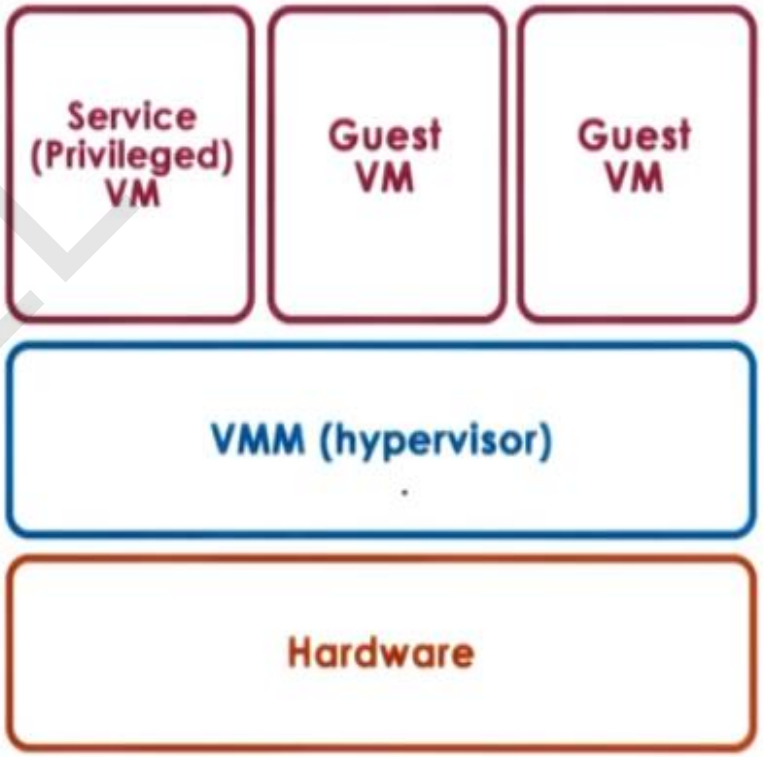
2. Hosted Hypervisor (Type 2)



Bare-metal virtualization model

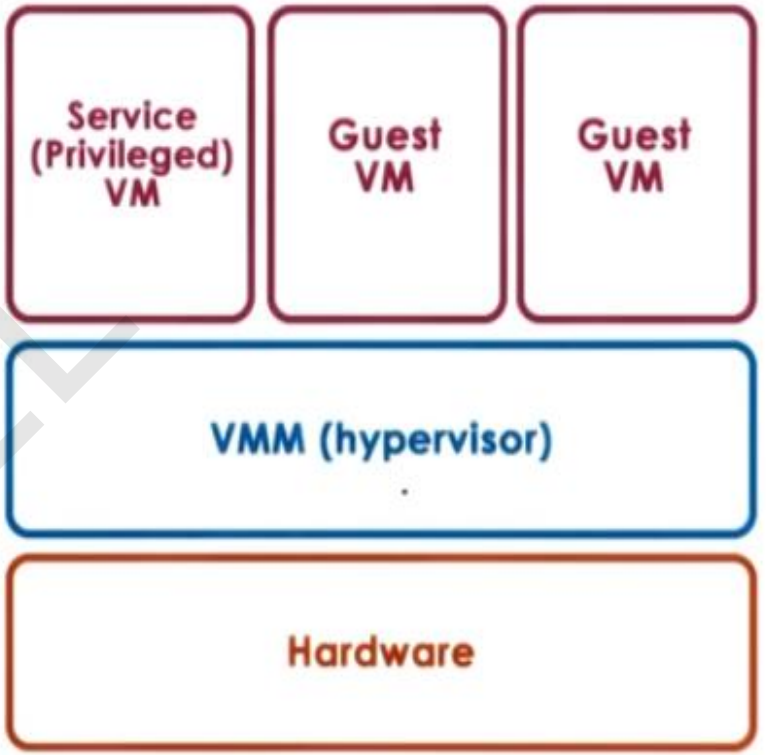
Bare-metal hypervisor (Type 1)

- **VMM (hypervisor)** manages all hardware resources and supports execution of entire VMs.
- **Privileged, service VM** to deal with devices (and other configuration and management task)



Bare-metal virtualization model

- This model is adapted by the **Xen** virtualization solution (open source or Citrix Xen Server) and also by the VMware's hypervisor, the **ESX hypervisor.**



(i) Xen (Open source or Citrix Xen Server)

- The VMs that are run in the virtualized environment are referred to as domains.
- The privileged domain is called dom 0, and the guest VMs are referred to as domUs.
- Xen is the actual hypervisor and all of the drivers are running in the privileged domain, in dom 0.

(ii) ESX (VMware)

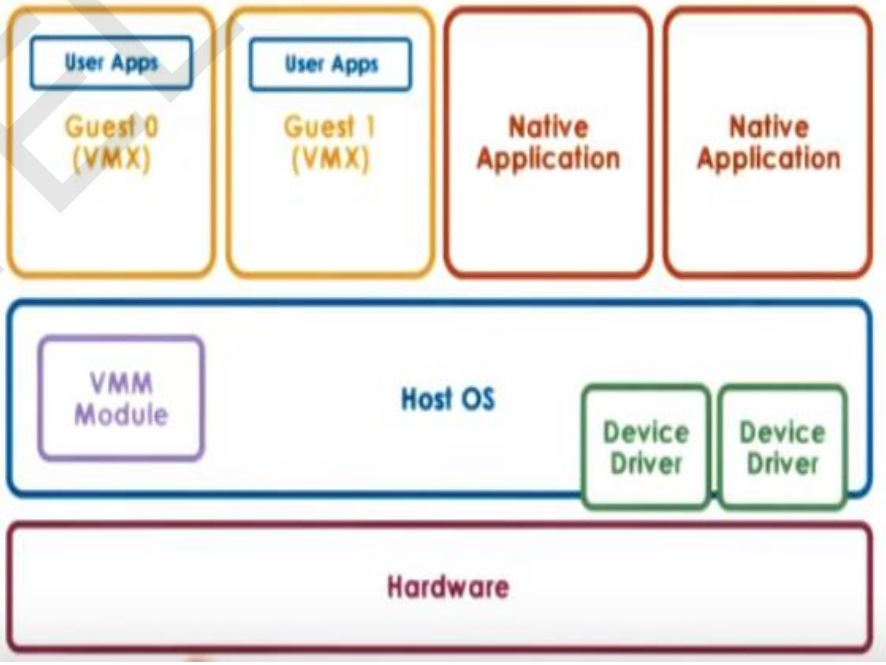
-Given that VMware and its hypervisors were first to market, VMware still owns the largest percentage of virtualized server cores. So these server cores run the ESX hypervisor and also provide the drivers for the different devices. That are going to be part of the hypervisor. To support a third party community of developers VMware exports a number of APIs.

Hosted virtualization model

Hosted Hypervisor (Type 2)

- In this model, at the lowest level, there is a **full fledged host OS that manages all of the hardware resources.**

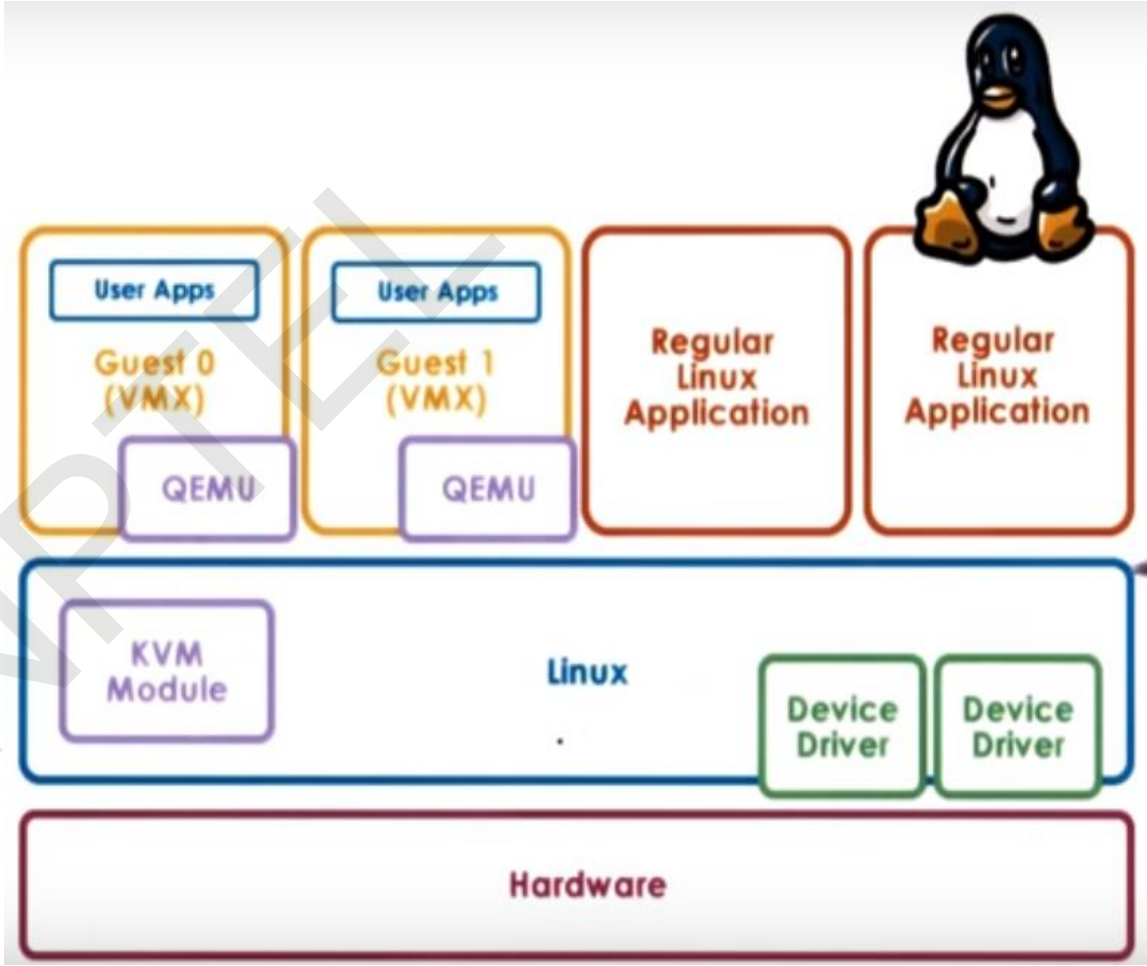
- The Host OS integrates a VMM module, that's responsible for providing the virtual machines with their virtual platform interface and for managing all of the context switching scheduling, etc.



Hosted virtualization model

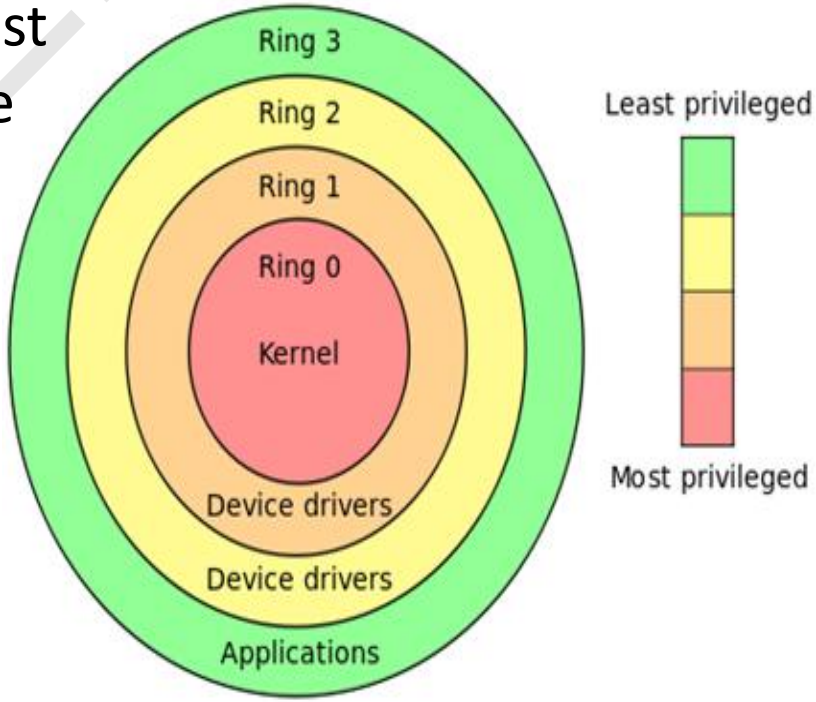
Example:

- **KVM** (Kernel-based VM)
- Based on Linux
- KVM kernel module + hardware emulator called QEMU for hardware virtualization
- Leverages large Linux open-source community



Hardware Protection Levels

- Commodity hardware actually has more than two protection levels.
- E.g. x86 architecture has four protection levels called **rings.**
- **Ring 3:** In contrast, ring 3 has the least level of privilege, so this is where the applications would reside.
- **Ring 0:** has the highest privilege and can access all of the resources and execute all hardware-supported instructions.



x86 Hardware without Virtualization

- The **x86** architecture offers **four levels** of privilege known as Ring 0, 1, 2 and 3 to operating systems and applications to manage access to the computer hardware. While user level applications typically run in Ring 3, the operating system needs to have direct access to the memory and hardware and must execute its privileged instructions in Ring 0.

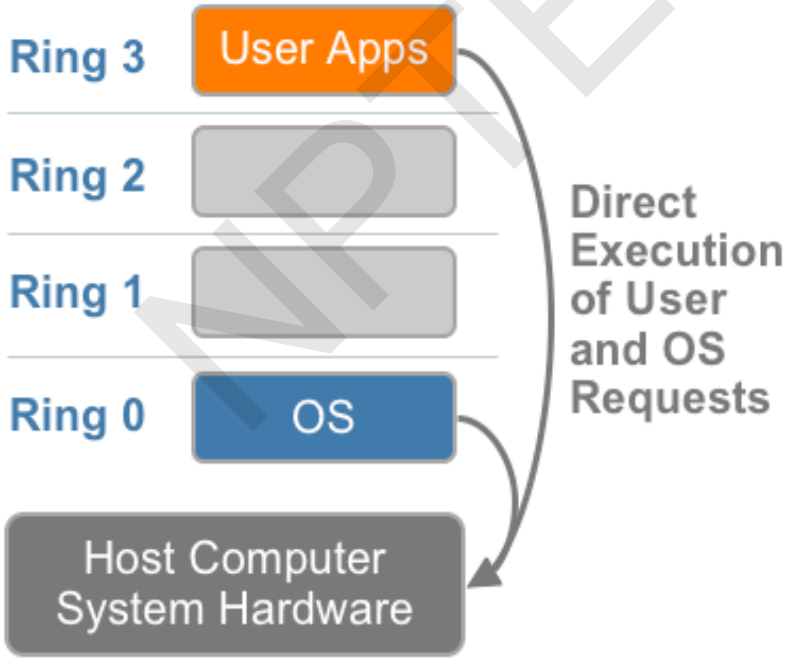


Fig - x86 privilege level architecture without virtualization

Processor Virtualization (Trap-and-Emulate)

- Guest instructions

  -executed **directly by hardware**

  -for non-privileged operations:
  hardware speeds=> efficiency

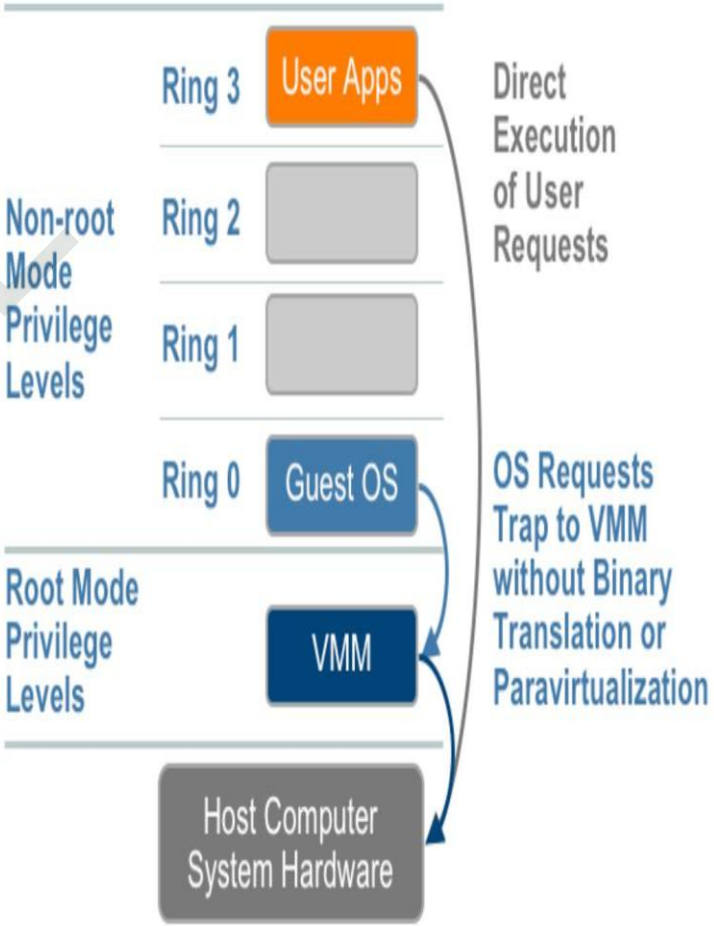
  -for privileged operations: trap to hypervisor

- Hypervisor determines what needs to be done:
- **If illegal operation:** terminate VM
- **If legal operation:** emulate the behavior the guest OS was expecting from the hardware

Problems with Trap-and-Emulate

- X85 Pre 2005
- -4 rings, no root/non-root modes yet
- -hypervisor in ring 0, guest OS in ring 1
- BUT: 17 privileged instructions do not trap! Fail silently!

  E.g. interrupt enable/disable bit in privileged register; POPF/PUSHF instructions that access it from ring 1 fail silently
- Hypervisor doesn't know, so it doesn't try to change settings
- OS doesn't know, so assumes change was successful/



Binary Translation

- **Main idea:** rewrite the VM binary to never issue those 17 instructions
- Pioneered by Mendel Rosenblum's group at Stanford, commercialized as Vmware
- Rosenblum awarded ACM Fellow for "reinventing virtualization"

Binary Translation

Binary translation:

- **Goal:** full virtualization=guest OS not modified
- **Approach:** dynamic binary translation

1. Inspect code blocks to be executed
2. If needed, translate to alternate instruction sequence

-e.g., to emulate desired behavior, possibly even avoiding trap

3. Otherwise, run at hardware speeds

-Cache translated blocks to amortize translation costs

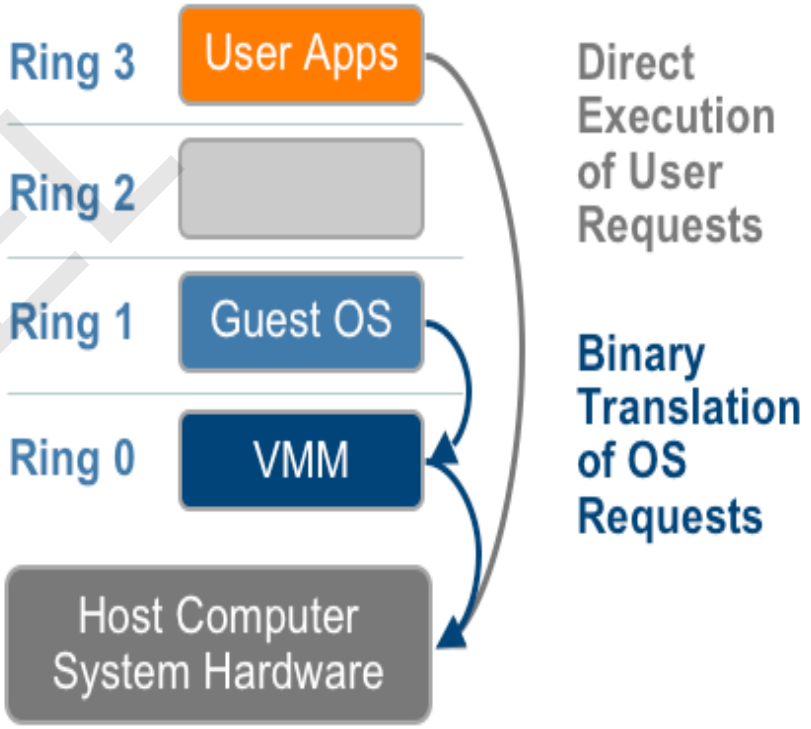
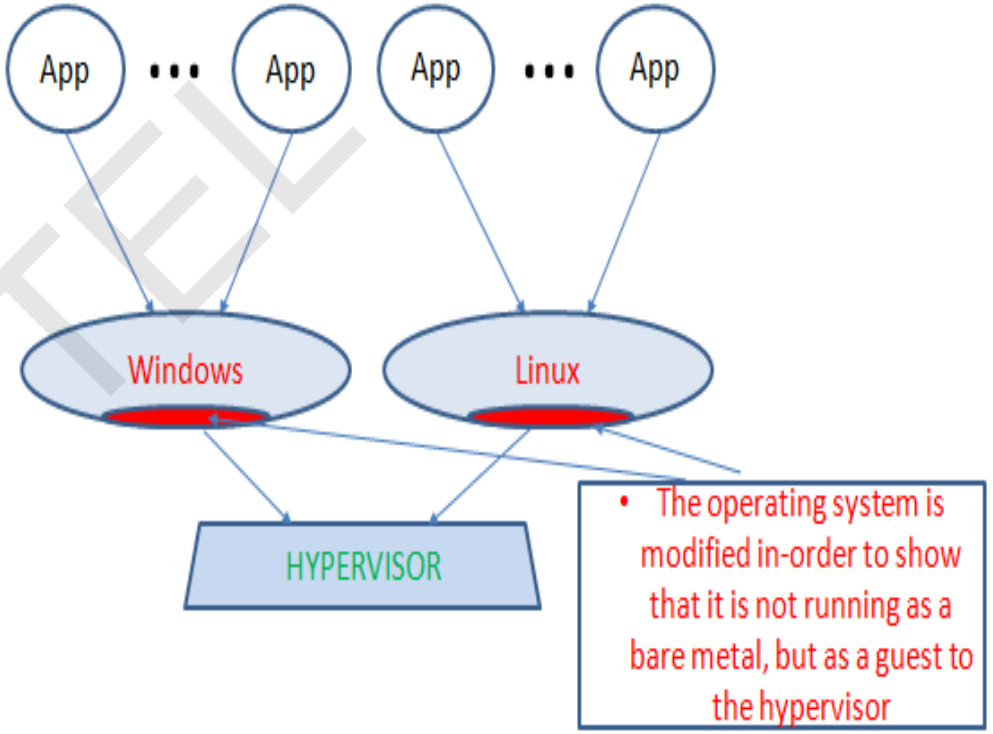


Fig.: Binary translation approach to x86 virtualization

Para virtualization

- **Goal:** performance; give up on unmodified guests
- Approach: Para virtualization =modify guest so that
- It knows it's running virtualized
- It makes explicit calls to the hypervisor (hypercalls)
- Hypercall (~system calls)
  - Package context info
  - Specify desired hypercall
  - Trap to VMM
  - e.g. Xen= open source hypervisor (XenSource -> Citrix)



Para virtualization: Review !

- What percentage of Guest OS code may need modification with para Virtualization?
  - 10 %
  - 50%
  - 30%
  - 2%
- Answer: less than 2%.
  - This can be shown by a proof-of-construction by XEN
  - Xen is a para virtualized hypervisor.

Memory Virtualization

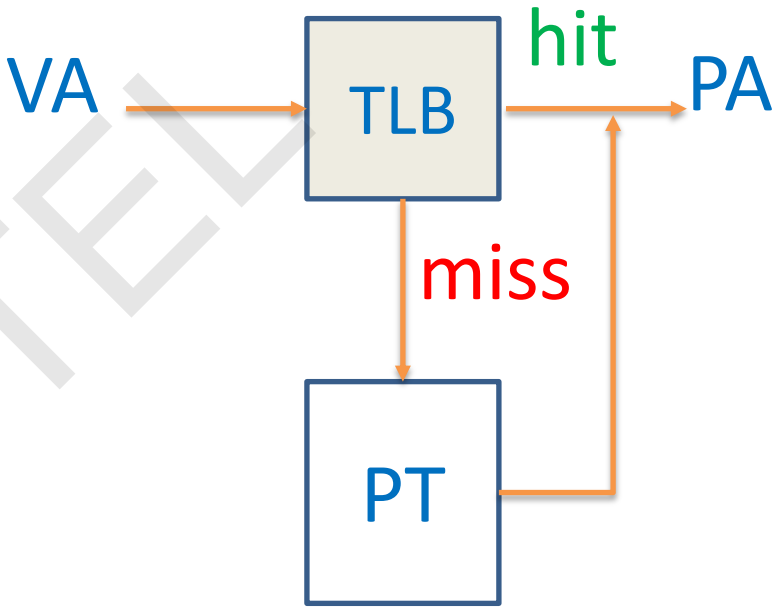
- To run multiple virtual machines on a single system, one has to virtualize the **MMU**(memory management unit) to support the guest OS.

- The **VMM** is responsible for mapping guest physical memory to the actual machine memory, and it uses **shadow page tables** to accelerate the mappings.

- The VMM uses **TLB** (translation lookaside buffer) hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.

Shadow Page table

- VA- Virtual Address
- PA- Physical Address



- CPU uses PT for address translation
- Hardware PT is really the S PT(shadow Page Table)

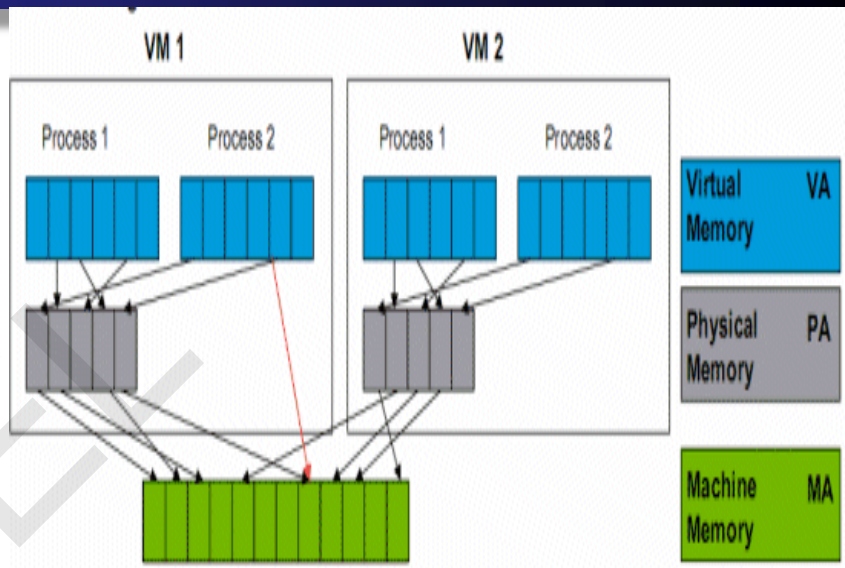
Memory Virtualization

- **Full virtualization**

  -all guests expect contiguous physical memory, starting at 0

  -virtual vs. physical vs. machine addresses and page frame numbers

  -Still leverages hardware MMU, TLB



Option 1:

-Guest page table: VM => PA

-Hypervisor: PA=> MA

-Too expensive

Option 2:

-Guest page table: VA => PA

-Hypervisor shadow Page Table: VA=> MA

-Hypervisor maintains consistence

e.g. invalidate on ctx switch, write-protect guest PT to track new mappings

Memory Virtualization

- **Para virtualization**

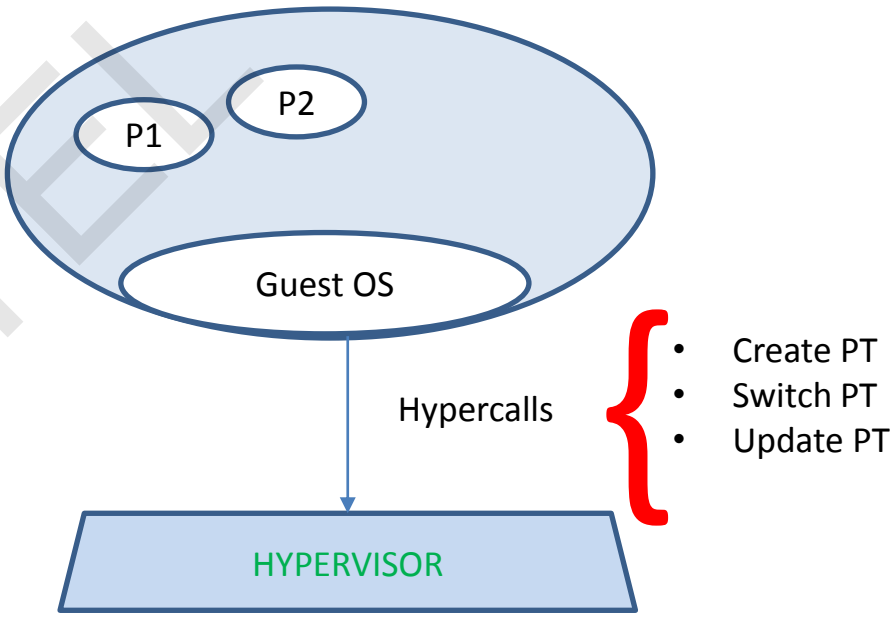
  -Guest aware of virtualization

  -No longer strict requirement on contiguous physical memory starting at 0

  -Explicitly registers page tables with hypervisor

  -Can “batch” page table updates to reduce VM exists

  -Other optimizations



- Overheads eliminated or reduced on newer platforms

Device Virtualization

- For CPUs and memory
- Less diversity, ISA (instruction set architectures)level standardization of interface
- For devices
- High diversity
- Lack of standard specification of device interface and behavior
- Three Key models for device virtualization

  (i) Passthrough Model

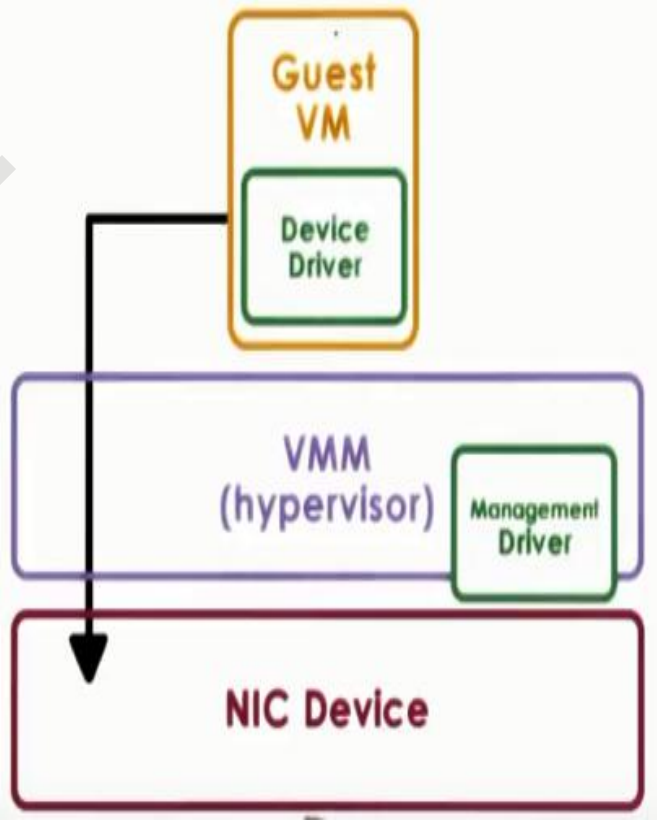
  (ii) Hypervisor Direct Model

  (iii) Split Device Driver Model

(i) Passthrough Model

Approach: VMM-level driver configures device access permissions

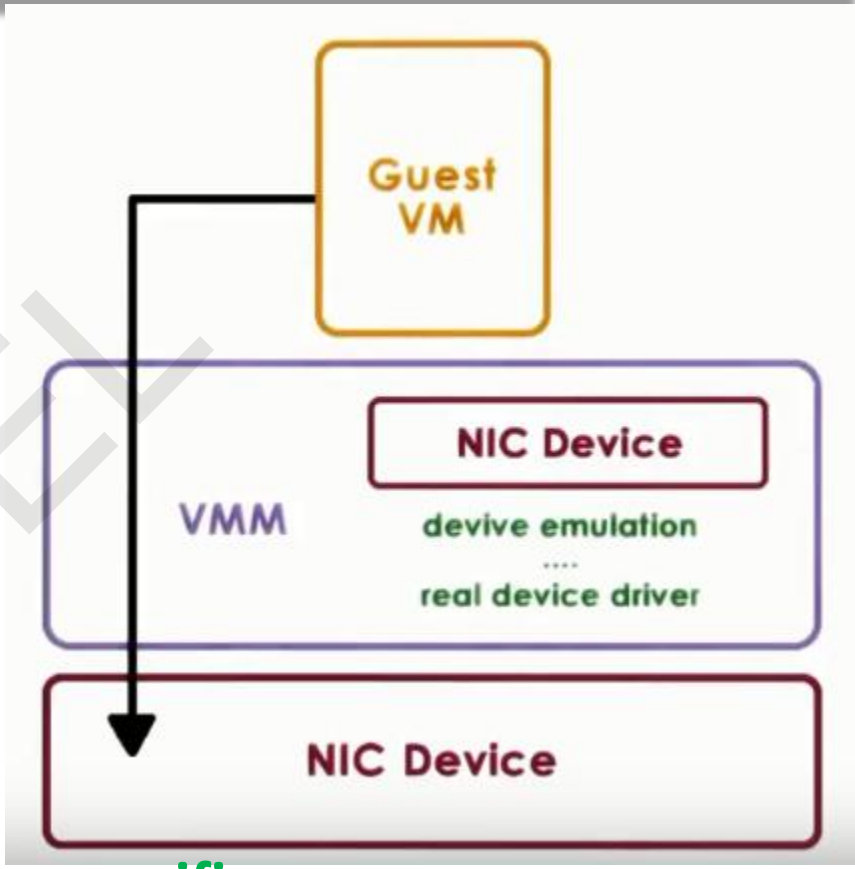
- VM provided with exclusive access to the device
- VM can directly access the device (also called VMM-bypass model)
- Device sharing will become difficult.
- VMM must have exact type of device as what VM expects
- VM migration is tricky



(ii) Hypervisor-Direct Model

Approach:

- VMM intercepts all device accesses
- Emulate device operation:
  - translate to generic I/O operation
  - traverse VMM-resident I/O stack
  - invoke VMM-resident driver



Key benefits:

VM decoupled from physical device

Sharing, migration, dealing with device specifies

Downside of the model

Latency of device operations

Device driver ecosystem complexities in hypervisor

(iii) Split-Device Driver Model

Approach:

Device access control **split** between

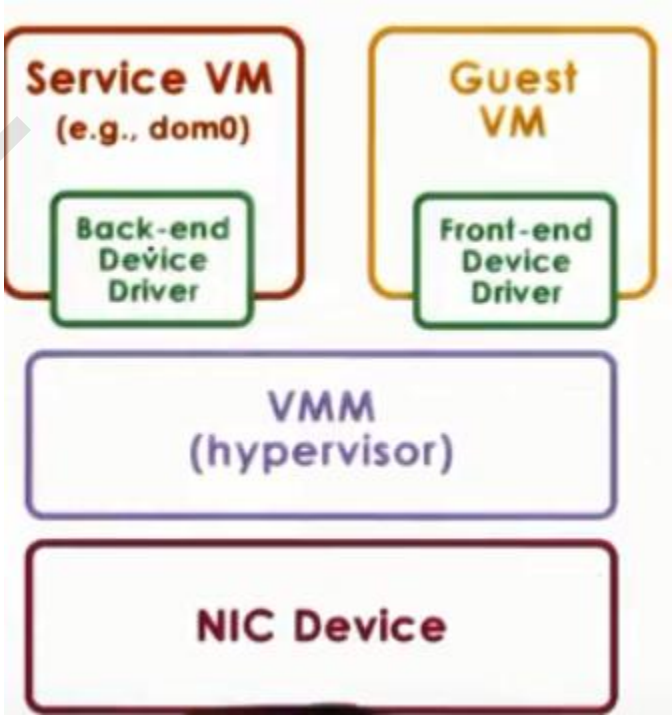
-Front end driver in guest VM (device API)

-Back-end driver in service VM (or host)

Modified guest drivers

i.e. Limited to para virtualized guests

Eliminate emulation overhead allow for better management of shared devices



Conclusion

- In this lecture, we have **defined virtualization** and discussed the main virtualization approaches.

- We have also described **processor virtualization, memory virtualization and device virtualization** used in virtualization solutions, such as Xen, KVM and the VMware .

Hotspot Mitigation for Virtual Machine Migration

Dr. Rajiv Misra

Associate Professor

Dept. of Computer Science & Engg.

Indian Institute of Technology Patna

rajivm@iitp.ac.in

Preface

Content of this Lecture:

- In this lecture, we will discuss ***hot spot mitigation techniques*** to automate the task of monitoring and detecting hotspots by determining a new mapping of physical to virtual resources and initiating the necessary migrations.

- We will discuss a ***black-box approach*** that is fully OS- and application-agnostic and a ***gray-box approach*** that exploits OS- and application-level statistics.

Enterprise Data Centers

- **Data Centers are composed of:**
  - Large clusters of servers
  - Network attached storage devices
- **Multiple applications per server**
  - Shared hosting environment
  - Multi-tier, may span multiple servers
- **Allocates resources to meet Service Level Agreements (SLAs)**
- **Virtualization increasingly common**

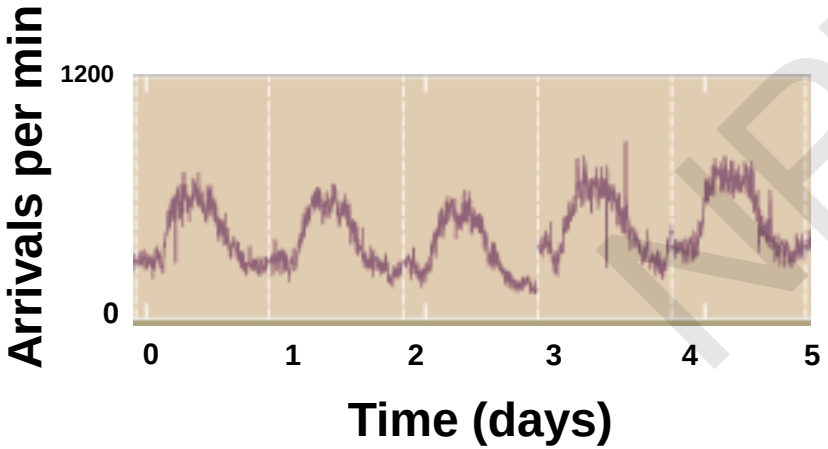
Benefits of Virtualization

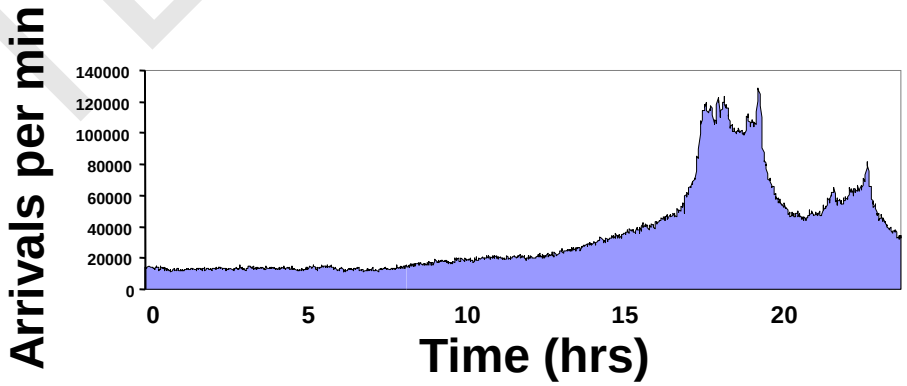
- **Run multiple applications on one server**
  - Each application runs in its own virtual machine
- **Maintains isolation**
  - Provides security
- **Rapidly adjust resource allocations**
  - CPU priority, memory allocation
- **VM migration**
  - "Transparent" to application
  - No downtime, but incurs overhead

How can we use virtualization to more efficiently utilize data center resources?

Data Center Workloads

- Web applications, enterprise systems, e-commerce sites etc see dynamic workloads
- dynamic workload fluctuations: caused by incremental growth, time-of-day effects, and flash crowds etc





How can we provision resources to meet these changing demands while meeting SLAs is a complex task**?**

Provisioning Methods

- **Hotspots** form if resource demand exceeds provisioned capacity

- **Over-provisioning**
  - Allocate for peak load
    - Wastes resources
    - Not suitable for dynamic workloads
    - Difficult to predict peak resource requirements

- **Dynamic provisioning**
  - Adjust based on workload
    - Often done manually
    - Becoming easier with virtualization

Problem Statement

How can we automatically (i) monitoring for resource usage,(ii) hotspot detection, and (iii) mitigation ie determining a new mapping and initiating the necessary migrations (ie **detect and mitigate Hotspots) in virtualized data centers?**

Hotspot Mitigation Problem

- Once a hotspot has been detected and new allocations have been determined for overloaded VMs, the migration manager invokes its hotspot mitigation algorithm.
- This algorithm determines which virtual servers to migrate and where in order to dissipate the hotspot.
- **Determining a new mapping of VMs to physical servers that avoids threshold violations is NP-hard—the multidimensional bin packing problem can be reduced to this problem, where each physical server is a bin with dimensions corresponding to its resource constraints and each VM is an object that needs to be packed with size equal to its resource requirements.**
- Even the problem of determining if a valid packing exists is NP-hard.

Research Challenges

- **Hotspot Mitigation:** automatically detect and mitigate hotspots through virtual machine migration
- ***When*** to migrate?
- ***Where*** to move to?
- ***How much*** of each resource to allocate?
- **How much information needed to make decisions?**

Sandpiper

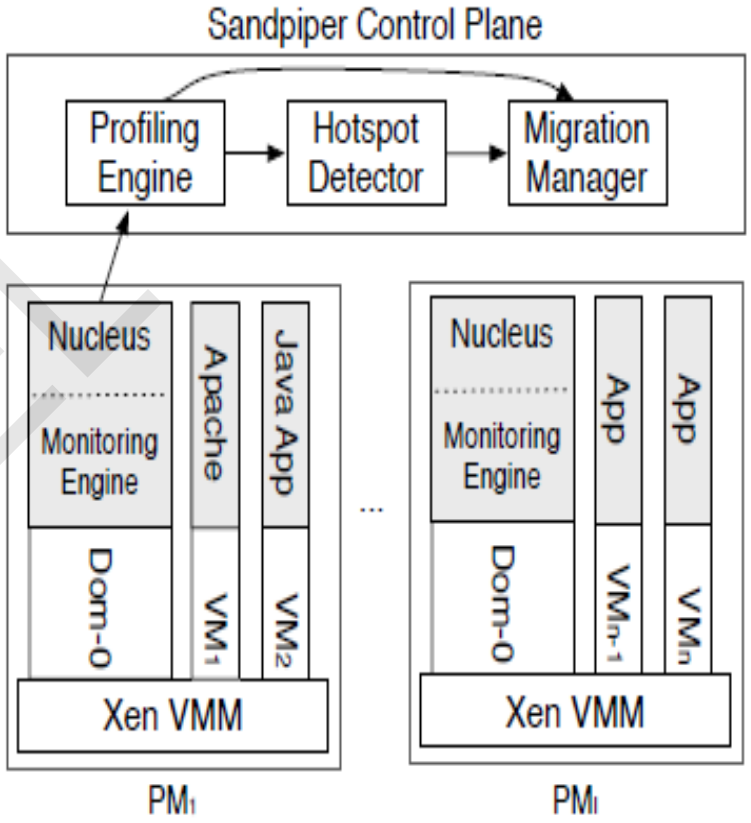
A migratory bird

Background

- **Dynamic replication**:
  - Dynamic provisioning approaches are focused on dynamic replication, where the number of servers allocated to an application is varied.
- **Dynamic slicing**:
  - In dynamic slicing, the fraction of a server allocated to an application is varied.
- **Application migration:**
  - In the virtualization, VM migration is performed for dynamic provisioning.
  - Migration is transparent to applications executing within virtual machines.

Sandipiper Architecture

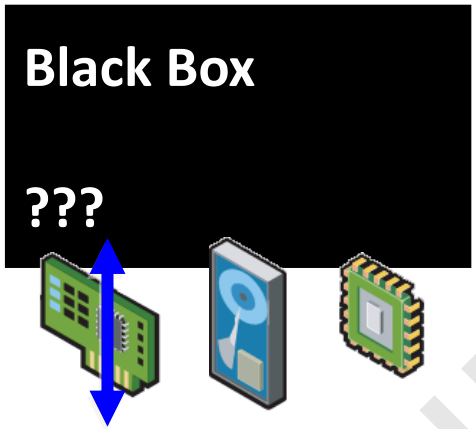
- **Nucleus**
  - *Monitor* resources
  - Report to control plane
  - One per server
  - Runs inside a special virtual server (domain 0 in Xen)
- **Control Plane**
  - Centralized server
- **Hotspot Detector**
  - Detect *when* a hotspot occurs
- **Profiling Engine**
  - Decide *how much* to allocate
- **Migration Manager**
  - Determine *where* to migrate

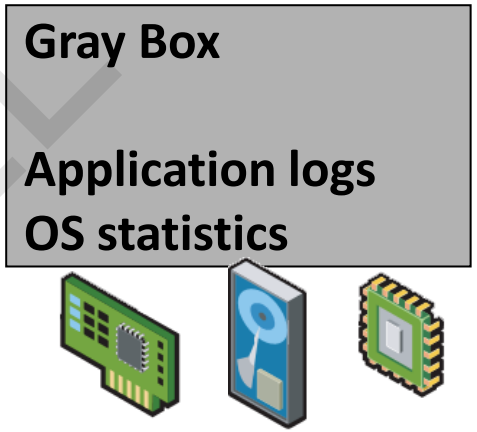


PM = Physical Machine
VM = Virtual Machine

Black-Box and Gray-Box

- **Black-box:** only data from outside the VM
  - Completely OS and application agnostic



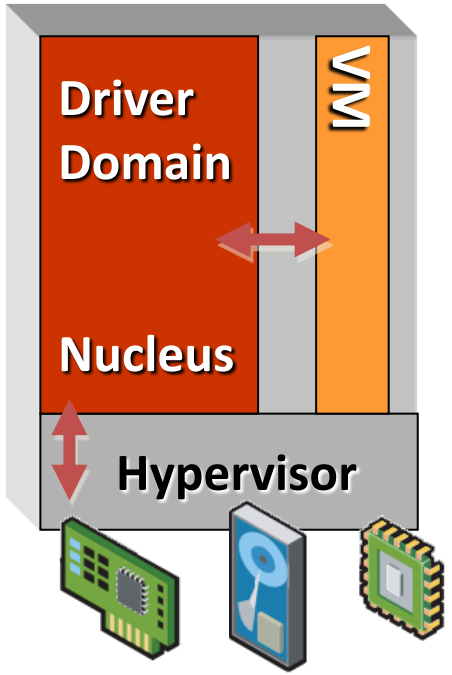


- **Gray-Box:** access to OS stats and application logs
  - Request level data can improve detection and profiling
  - Not always feasible – customer may control OS

Is black-box sufficient?
What do we gain from gray-box data?

Black-box Monitoring

- **Xen uses a "Driver Domain"**
  - Special VM with network and disk drivers
  - Nucleus runs here
- **CPU**
  - Scheduler statistics
- **Network**
  - Linux device information
- **Memory**
  - Detect swapping from disk I/O
  - Only know when performance is poor



Black-box Monitoring

- **CPU Monitoring:** By incrementing the Xen hypervisor, it is possible to provide domain-0 with access to CPU scheduling events which indicate when a VM is scheduled and when it relinquishes the CPU. These events are tracked to determine the duration for which each virtual machine is scheduled within each measurement interval *I*.

- **Network Monitoring:** Domain-0 in Xen implements the network interface driver and all other domains access the driver via clean device abstractions. Xen uses a **virtual firewall-router (VFR) interface**; each domain attaches one or more virtual interfaces to the VFR. Doing so enables Xen to multiplex all its virtual interfaces onto the underlying physical network interface.

Black-box Monitoring

- **Memory Monitoring:** Black-box monitoring of memory is challenging since Xen allocates a user specified amount of memory to each VM and requires the OS within the VM to manage that memory; as a result, the memory utilization is only known to the OS within each VM.

- It is possible to instrument Xen to observe memory accesses within each VM through the use of shadow page tables, which is used by Xen's migration mechanism to determine which pages are dirtied during migration.

- However, trapping each memory access results in a significant application slowdown and is only enabled during migrations. Thus, memory usage statistics are not directly available and must be inferred.

Black-box Monitoring

- **Black-box monitoring is useful in scenarios where it is not feasible to "peek inside" a VM to gather usage statistics. Hosting environments, for instance, run third-party applications, and in some cases, third-party installed OS distributions.**
- Amazon's Elastic Computing Cloud (EC2) service, for instance, provides a **"barebone"** virtual server where customers can load their own OS images.
- While OS instrumentation is not feasible in such environments, there are environments such as corporate data centers where both the hardware infrastructure and the applications are owned by the same entity.
- In such scenarios, it is feasible to gather OS-level statistics as well as application logs, which can potentially enhance the quality of decision making.

Gray-box Monitoring

- **Gray-box monitoring** can be supported, when feasible, using a light-weight monitoring daemon that is installed inside each virtual server.
- In Linux, the monitoring daemon uses the **/proc** interface to gather OS level statistics of CPU, network, and memory usage. The memory usage monitoring, in particular, enables proactive detection and mitigation of memory hotspots.
- The monitoring daemon also can process logs of applications such as web and database servers to derive statistics such as request rate, request drops and service times.
- Direct monitoring of such application-level statistics enables explicit detection of SLA violations, in contrast to the black-box approach that uses resource utilizations as a proxy metric for SLA monitoring.

What is a Hotspot ?

- **A hotspot indicates a resource deficit on the underlying physical server to service the collective workloads of resident VMs.**

- Before the hotspot can be resolved through migrations, The system must first estimate how much additional resources are needed by the overloaded VMs to fulfill their SLAs; these estimates are then used to locate servers that have sufficient idle resources.

Hotspot Detection

- **The hotspot detection algorithm** is responsible for signalling a need for VM migration whenever SLA violations are detected implicitly by the black-box approach or explicitly by the gray-box approach.
- Hotspot detection is performed on a per-physical server basis in the black-box approach—a hot-spot is flagged if the aggregate CPU or network utilizations on the physical server exceed a threshold or if the total swap activity exceeds a threshold.
- In contrast, explicit SLA violations must be detected on a per-virtual server basis in the gray-box approach—a hotspot is flagged if the memory utilization of the VM exceeds a threshold or if the response time or the request drop rate exceed the SLA-specified values.

Hotspot Detection

- To ensure that a small transient spike does not trigger needless migrations, a hotspot is flagged only if thresholds or SLAs are exceeded for a sustained time. Given a time-series profile, a hotspot is flagged if at least k out the n most recent observations as well as the next predicted value exceed a threshold. With this constraint, we can filter out transient spikes and avoid needless migrations.
- The values of k and n can be chosen to make hotspot detection aggressive or conservative. For a given n, small values of k cause aggressive hotspot detection, while large values of k imply a need for more sustained threshold violations and thus a more conservative approach.
- **In the extreme, n = k = 1** is the most aggressive approach that flags a hostpot as soon as the threshold is exceeded. Finally, the threshold itself also determines how aggressively hotspots are flagged; lower thresholds imply more aggressive migrations at the expense of lower server utilizations, while higher thresholds imply higher utilizations with the risk of potentially higher SLA violations.

Hotspot Detection

- In addition to requiring **k out of n violations**, we also require that the next predicted value exceed the threshold.

- The additional requirement ensures that the hotspot is likely to persist in the future based on current observed trends. Also, predictions capture rising trends, while preventing declining ones from triggering a migration.

Hotspot Detection

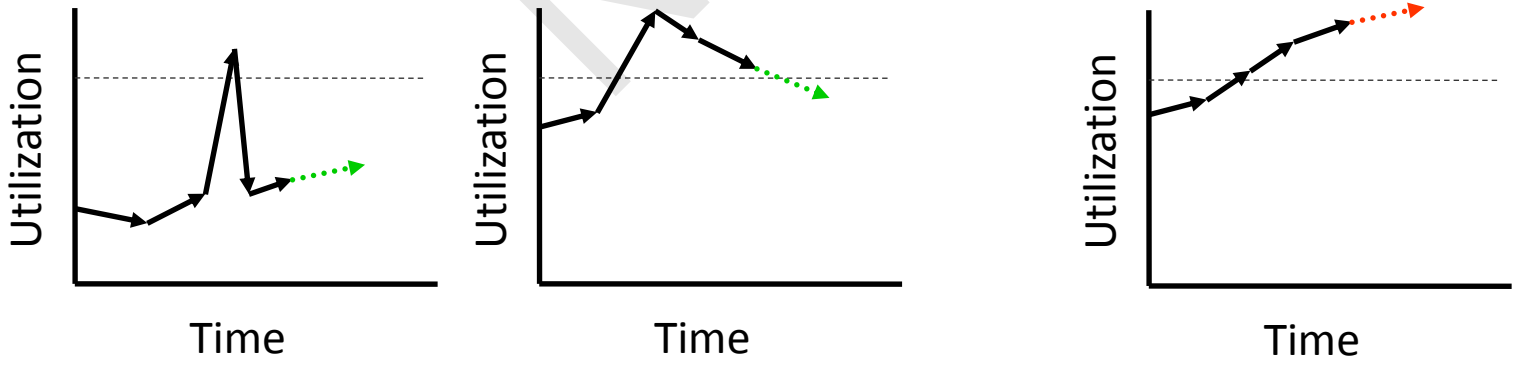
- It employs **time-series prediction techniques to predict future values.** Specifically, the system relies on the auto-regressive family of predictors, where the n-th order predictor AR(n) uses n prior observations in conjunction with other statistics of the time series to make a prediction. To illustrate the first-order AR(1) predictor, consider a sequence of observations: **u1, u2, ..., uk**. Given this time series, we wish to predict the demand in the **(k + 1)th** interval. Then the first-order **AR(1)** predictor makes a prediction using the previous value **uk**, the mean of the time series values μ, and the parameter which captures the variations in the time series. The prediction ˆuk+1 is given by:

$$\hat{u}_{k+1} = \mu + \phi(u_k - \mu)$$

- As new observations arrive from the nuclei, the hot spot detector updates its predictions and performs the above checks to flag new hotspots in the system.

Hotspot Detection – When?

- **Resource Thresholds**
  - Potential hotspot if utilization exceeds threshold
- **Only trigger for *sustained* overload**
  - Must be overloaded for k out of n measurements
- **Autoregressive Time Series Model**
  - Use historical data to predict future values
  - Minimize impact of transient spikes



Not overloaded

Hotspot Detected!

Resource Provisioning: Black-box Provisioning

- The provisioning component needs to estimate the peak CPU, network and memory requirement of each overloaded VM; doing so ensures that the SLAs are not violated even in the presence of peak workloads.

- **Estimating peak CPU and network bandwidth needs:** Distribution profiles are used to estimate the peak CPU and network bandwidth needs of each VM. The tail of the usage distribution represents the peak usage over the recent past and is used as an estimate of future peak needs.

- This is achieved by computing a high percentile **(e.g., the 95th percentile)** of the CPU and network bandwidth distribution as an initial estimate of the peak needs.

Limitation of the black-box approach

- **Example:** Consider two virtual machines that are assigned CPU weights of 1:1 resulting in a fair share of 50% each. Assume that VM1 is overloaded and requires 70% of the CPU to meet its peak needs. If VM2 is underloaded and only using 20% of the CPU, then the work conserving Xen scheduler will allocate 70% to VM1.

- In this case, the tail of the observed distribution is a good indicator of VM1's peak need. In contrast, if VM2 is using its entire fair share of 50%, then VM1 will be allocated exactly its fair share. In this case, the peak observed usage will be 50%, an underestimate of the actual peak need. Since the system can detect that the CPU is fully utilized, it will estimate the peak to be 50 +Δ .

Resource Provisioning: (i) Black-box Provisioning

- **Estimating peak memory needs:** Xen allows a fixed amount of physical memory to be assigned to each resident VM; this allocation represents a hard upper-bound that can not be exceeded regardless of memory demand and regardless of the memory usage in other VMs.

- Consequently, the techniques for estimating the peak CPU and network usage do not apply to memory. The provisioning component uses observed swap activity to determine if the current memory allocation of the VM should be increased.

- If swap activity exceeds the threshold indicating memory pressure, then the current allocation is deemed insufficient and is increased by a constant amount Δm.

Resource Provisioning: (ii) Gray-box Provisioning

- **Since the gray-box approach has access to application level logs, information contained in the logs can be utilized to estimate the peak resource needs of the application.**
- Unlike the black-box approach, the peak needs can be estimated even when the resource is fully utilized.
- To estimate peak needs, the peak request arrival rate is first estimated. Since the number of serviced requests as well as the number of dropped requests are typically logged, the incoming request rate is the summation of these two quantities.
- Given the distribution profile of the arrival rate, the peak rate is simply a high percentile of the distribution. Let λ_{peak} denote the estimated peak arrival rate for the application.

Estimating peak CPU needs:

- An application model is necessary to estimate the peak CPU needs. Applications such as web and database servers can be modeled as **G/G/1 queuing systems**. The behavior of such a **G/G/1 queuing system** can be captured using the following queuing theory result:

$$\lambda_{cap} \geq \left[s + \frac{\sigma_a^2 + \sigma_b^2}{2 \cdot (d - s)} \right]^{-1}$$

- where d is the mean response time of requests, s is the mean service time, and λ_{cap} is the request arrival rate. σ^2_a **and** σ^2_b are the variance of inter-arrival time and the variance of service time, respectively. Note that response time includes the full queueing delay, while service time only reflects the time spent actively processing a request.

Estimating peak CPU needs:

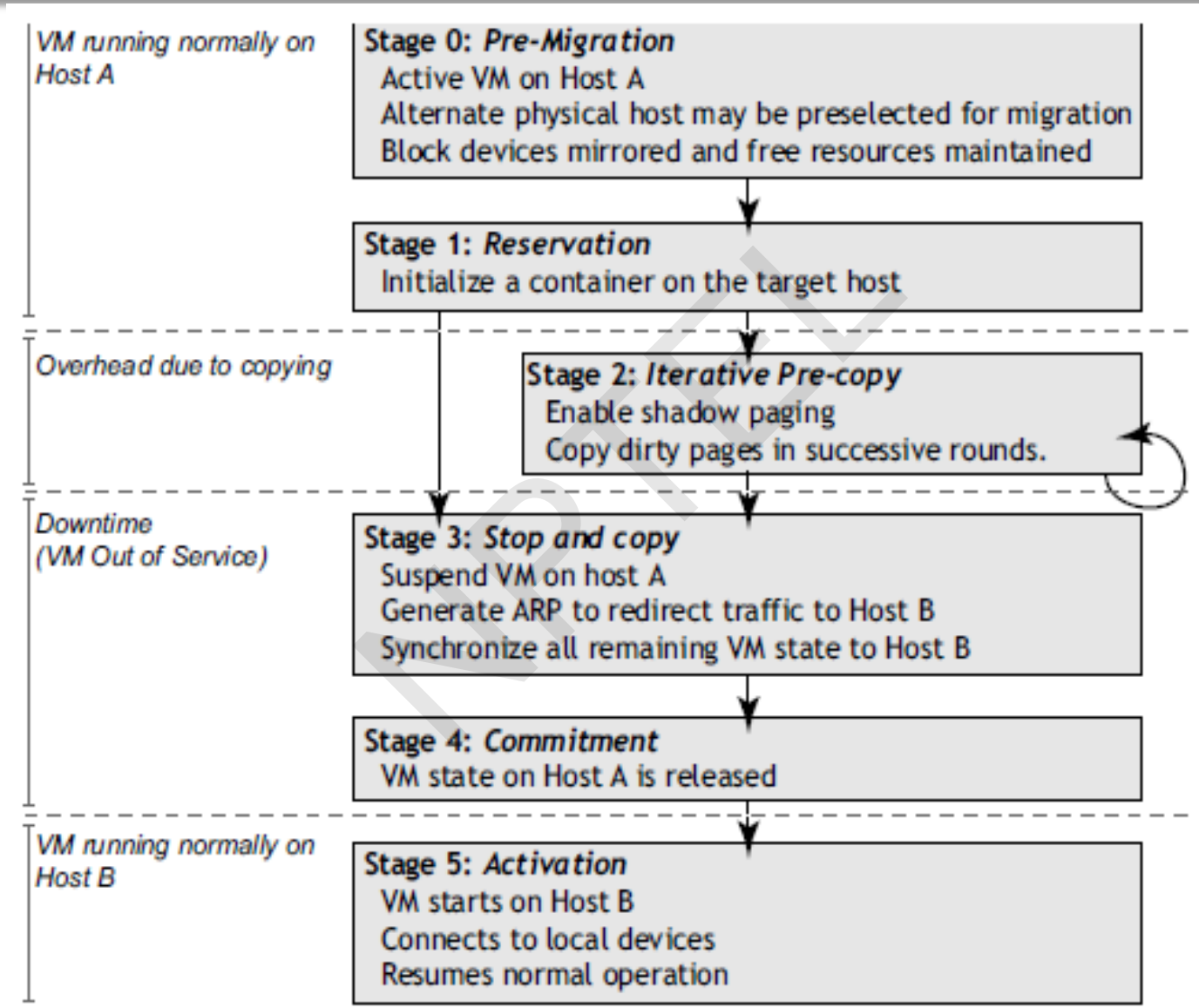
- While the desired response time d is specified by the SLA, the service time **s** of requests as well as the variance of inter-arrival and service times σ^2_a and σ^2_b can be determined from the server logs. By substituting these values into equation, a lower bound on request rate λ_{cap} that can be serviced by the virtual server is obtained.
- Thus, λ_{cap} represents the current capacity of the VM. To service the estimated peak workload λ_{peak} , the current CPU capacity needs to be scaled by the factor $\lambda_{peak} / \lambda_{cap}$
- Observe that this factor will be greater than 1 if the peak arrival rate exceeds the currently provisioned capacity. Thus, if the VM is currently assigned a CPU weight w, its allocated share needs to be scaled up by the factor $\lambda_{peak} / \lambda_{cap}$ to service the peak workload.

Estimating peak network needs:

- The peak network bandwidth usage is simply estimated as the product of the estimated peak arrival rate λ_{peak} and

- The mean requested file size **b**; this is the amount of data transferred over the network to service the peak workload. The mean request size can be computed from the server logs.

Live VM Migration Stages



Live VM Migration Stages

- **Stage 0: Pre-Migration** We begin with an active VM on physical host A. To speed any future migration, a target host may be preselected where the resources required to receive migration will be guaranteed.

- **Stage 1: Reservation** A request is issued to migrate an OS from host A to host B. We initially confirm that the necessary resources are available on B and reserve a VM container of that size. Failure to secure resources here means that the VM simply continues to run on A unaffected.

- **Stage 2: Iterative Pre-Copy** During the first iteration, all pages are transferred from A to B. Subsequent iterations copy only those pages dirtied during the previous transfer phase.

Live VM Migration Stages

- **Stage 3: Stop-and-Copy** We suspend the running OS instance at A and redirect its network traffic to B. As described earlier, CPU state and any remaining inconsistent memory pages are then transferred. At the end of this stage there is a consistent suspended copy of the VM at both A and B. The copy at A is still considered to be primary and is resumed in case of failure.

- **Stage 4: Commitment** Host B indicates to A that it has successfully received a consistent OS image. Host A acknowledges this message as commitment of the migration transaction: host A may now discard the original VM, and host B becomes the primary host.

- **Stage 5: Activation** The migrated VM on B is now activated. Post-migration code runs to reattach device drivers to the new machine and advertise moved IP addresses.

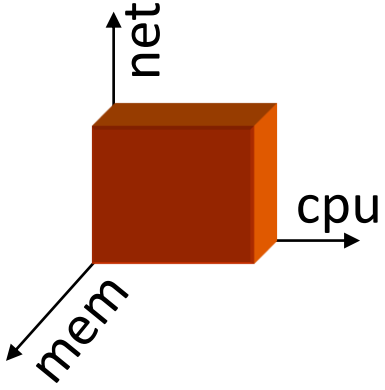
Hotspot Mitigation

- **The hotspot mitigation algorithm resorts to a heuristic to determine which overloaded VMs to migrate and where such that migration overhead is minimized.**
- Reducing the migration overhead (i.e., the amount of data transferred) is important, since Xen's live migration mechanism works by iteratively copying the memory image of the VM to the destination while keeping track of which pages are being dirtied and need to be resent. This requires Xen to intercept all memory accesses for the migrating domain, which significantly impacts the performance of the application inside the VM.
- **By reducing the amount of data copied over the network, It can minimize the total migration time, and thus, the performance impact on applications.**

Determining Placement – Where to?

- **Migrate VMs from overloaded to underloaded servers**

$$Volume = \frac{1}{1\text{-cpu}} * \frac{1}{1\text{-net}} * \frac{1}{1\text{-mem}}$$



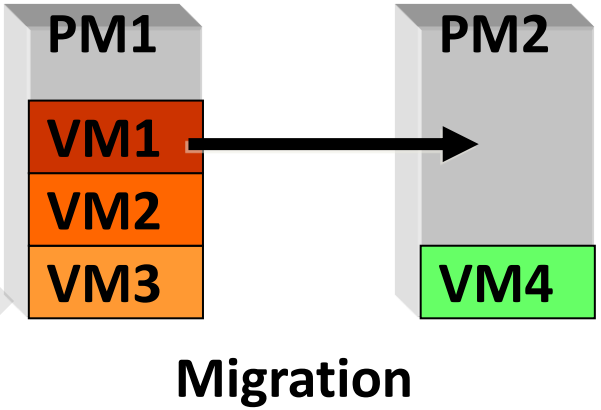
- **Use *Volume* to find most loaded servers**
  - Captures load on multiple resource dimensions
  - Highly loaded servers are targeted first

- **Migrations incur overhead**
  - Migration cost determined by RAM
  - Migrate the VM with highest Volume/RAM ratio

Maximize the amount of load transferred while minimizing the overhead of migrations

Placement Algorithm

- First try **migrations**
  - Displace VMs from high *Volume* servers
  - Use *Volume/RAM* to minimize overhead

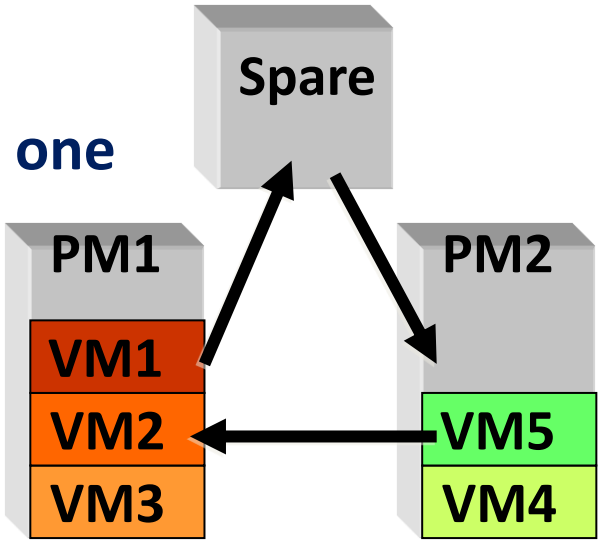


Migration

- **Don't create new hotspots!**
  - **What if high average load in system?**

- **Swap** if necessary
  - Swap a **high *Volume* VM** for **a low *Volume* one**
  - Requires 3 migrations
    - Can't support both at once

Swaps increase the number of hotspots we can resolve



Swap

Reading

USENIX NSDI '07

Black-box and Gray-box Strategies for Virtual Machine Migration

Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif†
Univ. of Massachusetts Amherst †*Intel, Portland*

Source: https://www.usenix.org/legacy/event/nsdi07/tech/full_papers/wood/wood.pdf

USENIX NSDI '05

Live Migration of Virtual Machines

Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen†,
Eric Jul†, Christian Limpach, Ian Pratt, Andrew Warfield
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge, UK
firstname.lastname@cl.cam.ac.uk
† *Department of Computer Science*
University of Copenhagen, Denmark
{jacobg,eric}@diku.dk

Source: https://www.usenix.org/legacy/event/nsdi05/tech/full_papers/clark/clark.pdf

Conclusion

- **Virtual Machine migration is a viable tool for dynamic data center provisioning.**

- In this lecture, we have discussed an approach to rapidly detect and eliminate hotspots while treating each VM as a **black-box.**

- **Gray-Box** information can improve performance in some scenarios
  - Proactive memory allocations

- **Future work**
  - Improved black-box memory monitoring
  - Support for replicated services