



**IMT Atlantique**

Bretagne-Pays de la Loire  
École Mines-Télécom

# CLOUD COMPUTING

# SOMMAIRE

1. INTRODUCTION
2. CLOUD PRINCIPLES
3. DATA CENTER -  
INFRASTRUCTURE
4. INSIDE THE CLOUD  
MODEL
5. INFRASTRUCTURE AS A  
SERVICE
6. SOFTWARE AS A SERVICE
7. CONCLUSION



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

# INTRODUCTION



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

## Jean-Marc Menaud

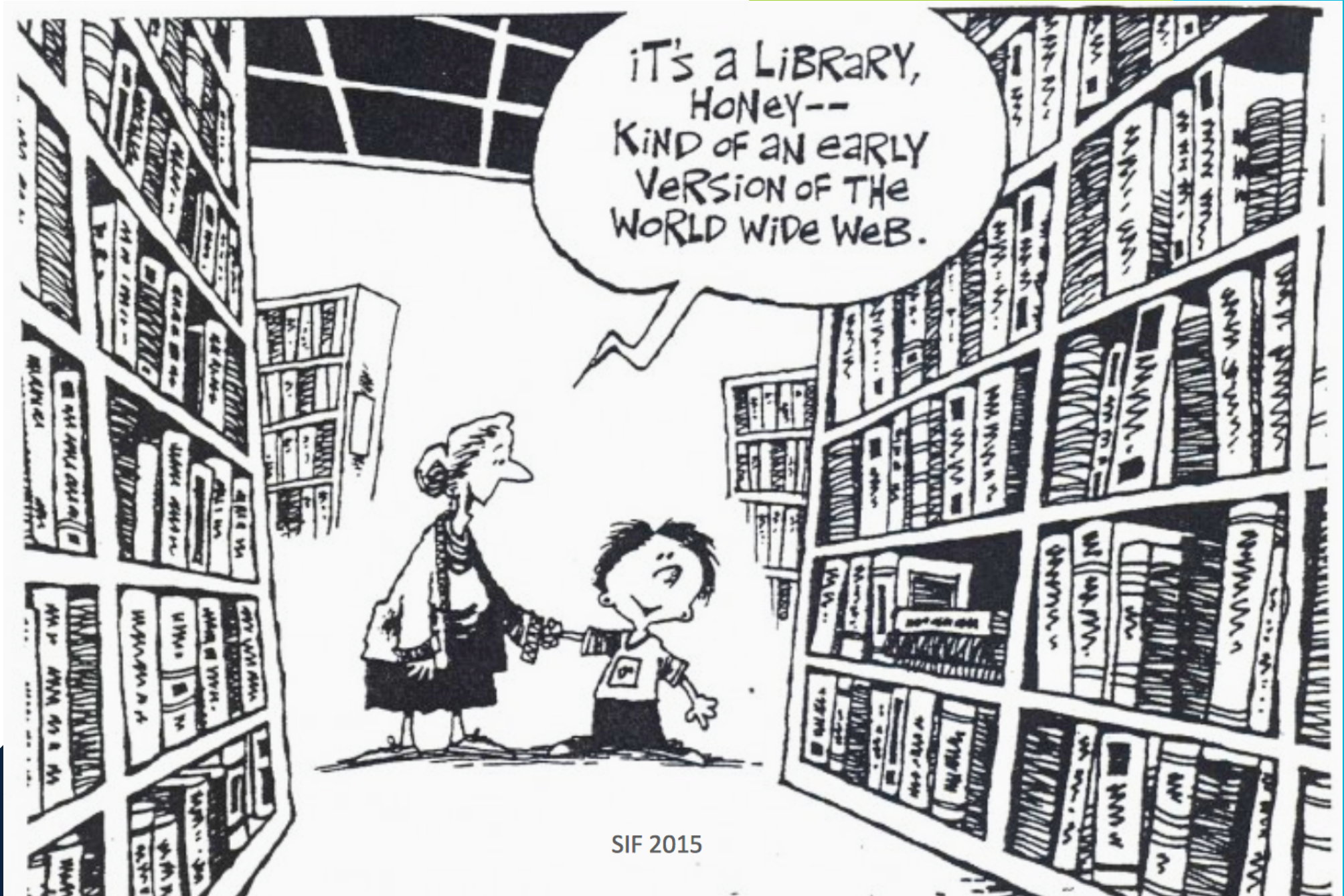
### ■ Full 1st Class Professor at IMT Atlantique

- Energy consumption optimization in large-scale distributed system
- **Data Center, Virtualization, Dynamic Consolidation, Workload driven, Power driven...**

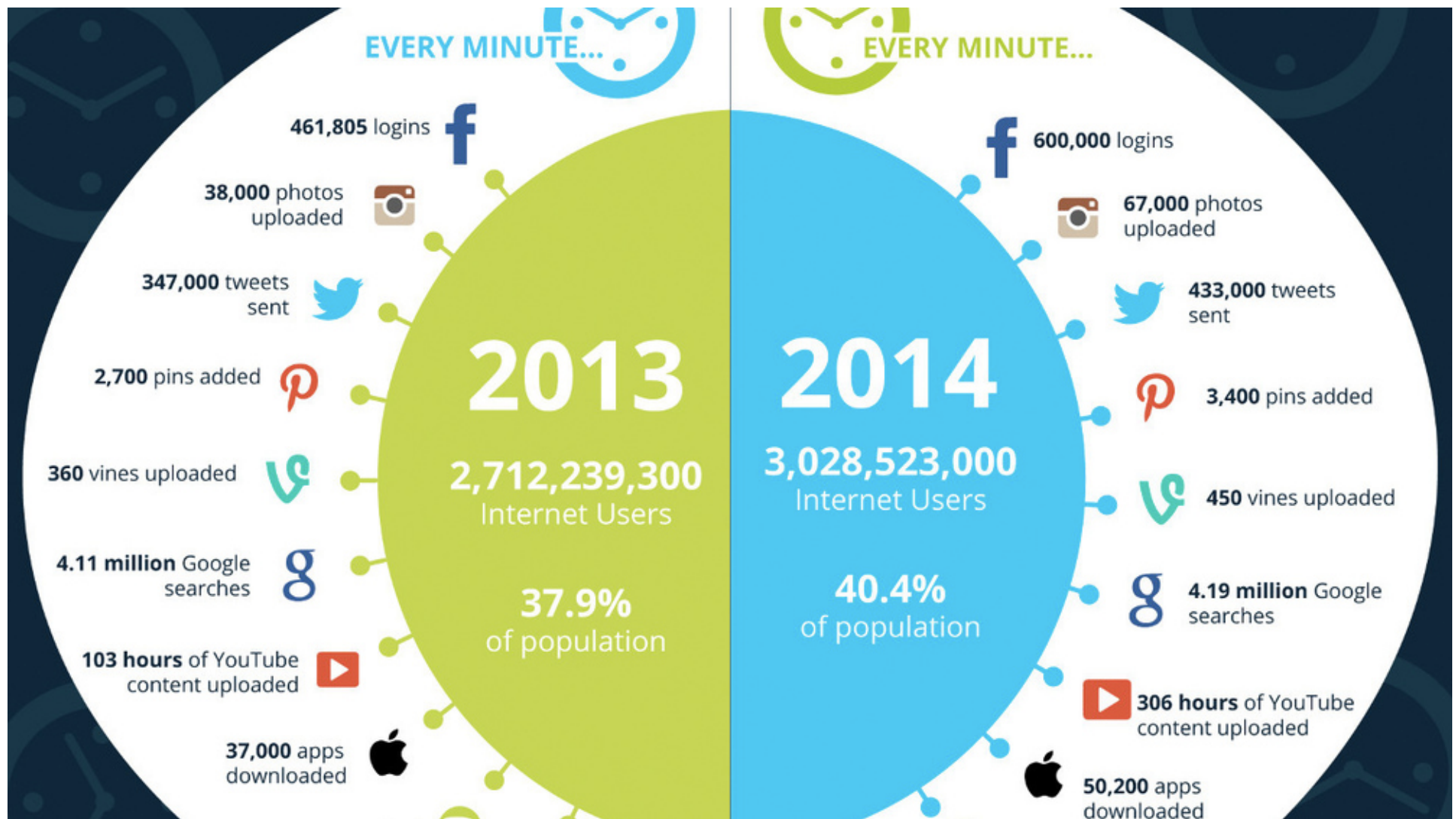
### ■ Co-Founder of EasyVirt

- French SME on Management solution for virtualized Data Center.





SIF 2015



# CHAPTER 1

# CLOUD PRINCIPLES



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom



**Cloud Computing** is a general term used to describe a new class of network based computing that takes place over the Internet,

- basically a step on from Utility Computing
- a collection/group of integrated and networked hardware, software and Internet infrastructure (called a platform).
- Using the Internet for communication and transport provides hardware, software and networking services to clients

These platforms hide the complexity and details of the underlying infrastructure from users and applications by providing very simple graphical interface or API (Applications Programming Interface).

- In addition, the platform provides on demand services, that are always on, anywhere, anytime and any place.
- Pay for use and as needed, elastic
  - scale up and down in capacity and functionalities
- The hardware and software services are available to
  - general public, enterprises, corporations and businesses markets

Cloud computing is an umbrella term used to refer to Internet based development and services

A number of characteristics define cloud data, applications services and infrastructure:

**Remotely hosted:** Services or data are hosted on remote infrastructure.

**Ubiquitous:** Services or data are available from anywhere.

**Commodified:** The result is a utility computing model similar to traditional that of traditional utilities, like gas and electricity - you pay for what you would want!



In 1997, Steve Jobs introduced his personal cloud based data management system and the efficiency of accessing data from any computer in the world. He outlines a vision for the future where cloud based computing will create safe systems free of hard drive failures and loss of data.

During a WWDC Q&A session with developers.



Steve Wozniak

I really worry about everything going to the cloud, I think it's going to be horrendous. I think there are going to be a lot of horrible problems in the next five years.

A lot of people feel, 'Oh, everything is really on my computer,' but I say the more we transfer everything onto the web, onto the cloud, the less we're going to have control over it.

(August 2012)





Richard Stallman

He said that cloud computing was simply a trap aimed at forcing more people to buy into locked, proprietary systems that would cost them more and more over time.

It's stupidity. It's worse than stupidity: it's a marketing hype campaign,

Somebody is saying this is inevitable – and whenever you hear somebody saying that, it's very likely to be a set of businesses campaigning to make it true.

(September 2008)

- Cloud computing enables companies and applications, which are system infrastructure dependent, to be **infrastructure-less**.
- By using the Cloud infrastructure on “pay as used and on demand”, all of us can **save in capital and operational investment!**
- Clients can:
  - Put their data on the platform instead of on their own desktop PCs and/or on their own servers.
  - They can put their applications on the cloud and use the servers within the cloud to do processing and data manipulations etc.

Cloud are transparent to users and applications, they can be built in multiple ways

branded products, proprietary open source, hardware or software, or just off-the-shelf PCs.

In general, they are built on **clusters of PC servers** and off-the-shelf components plus Open Source software combined with in-house applications and/or system software.

# CHAPTER 2

## DATA CENTER - INFRASTRUCTURE



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom





SIF 2015



1U



2U



4U

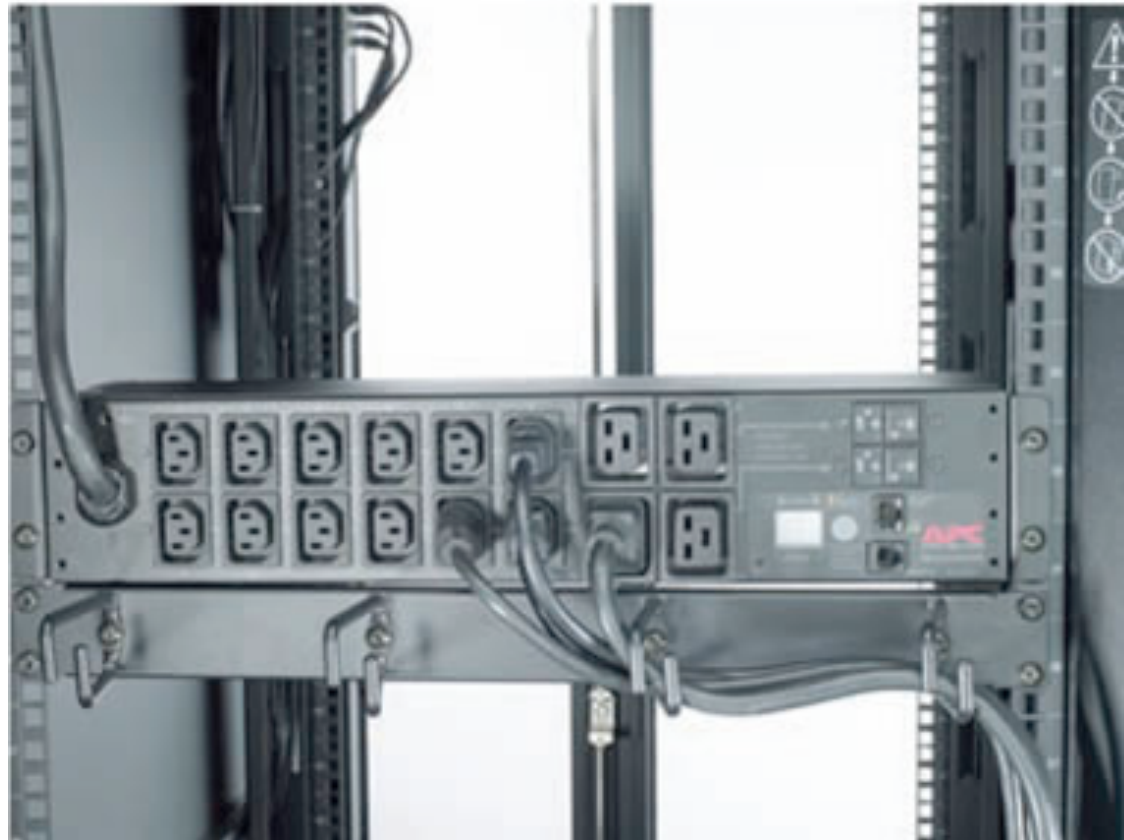


Source : [dell.com](http://dell.com)





Source : [server-racks.com](http://server-racks.com)



Source : 42u.com



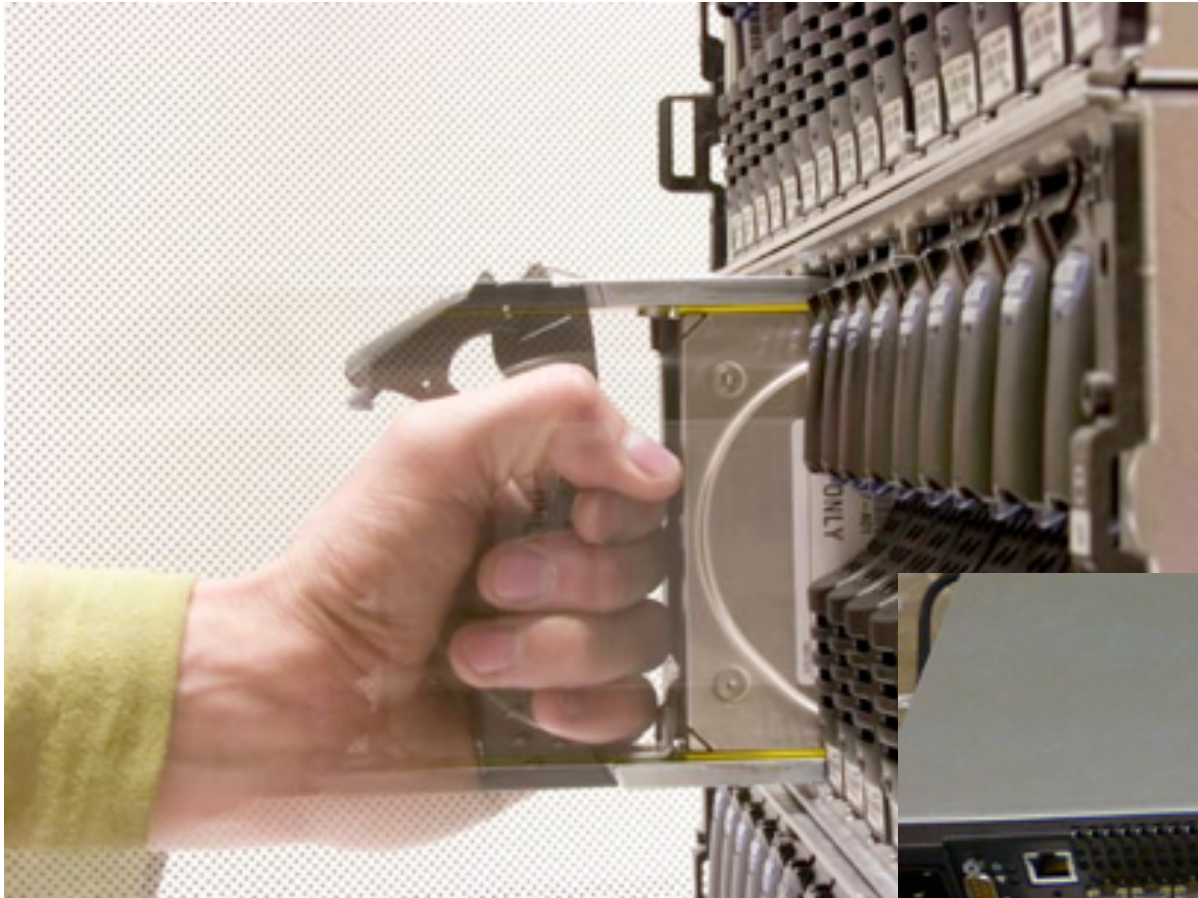


Source : cloudtp.com



*Switch*

Source : webpage.pace.edu

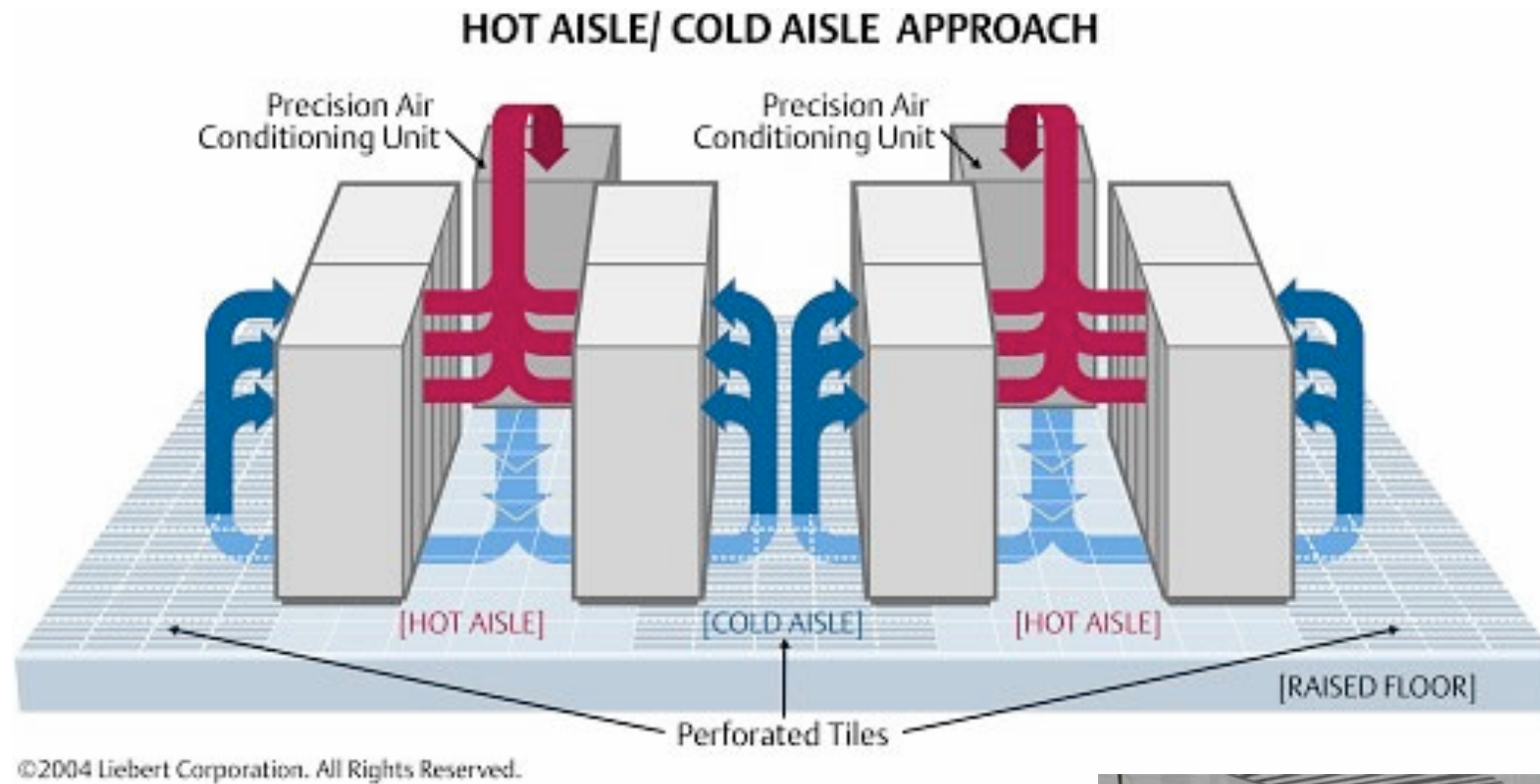


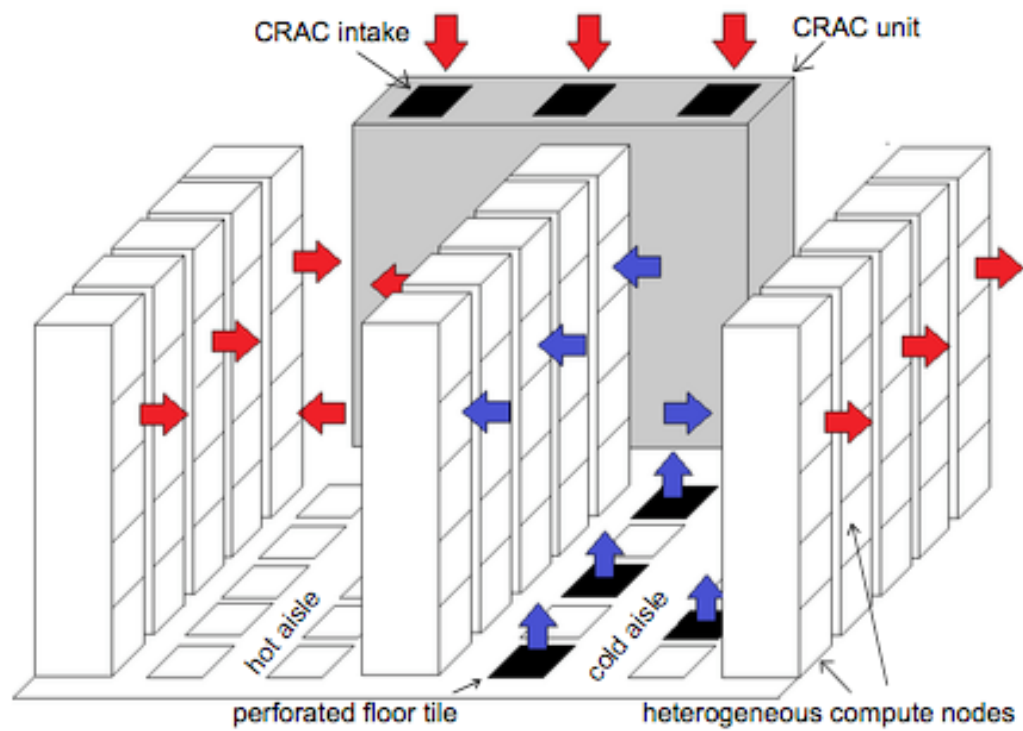
Source : [commons.wikimedia.org](https://commons.wikimedia.org)

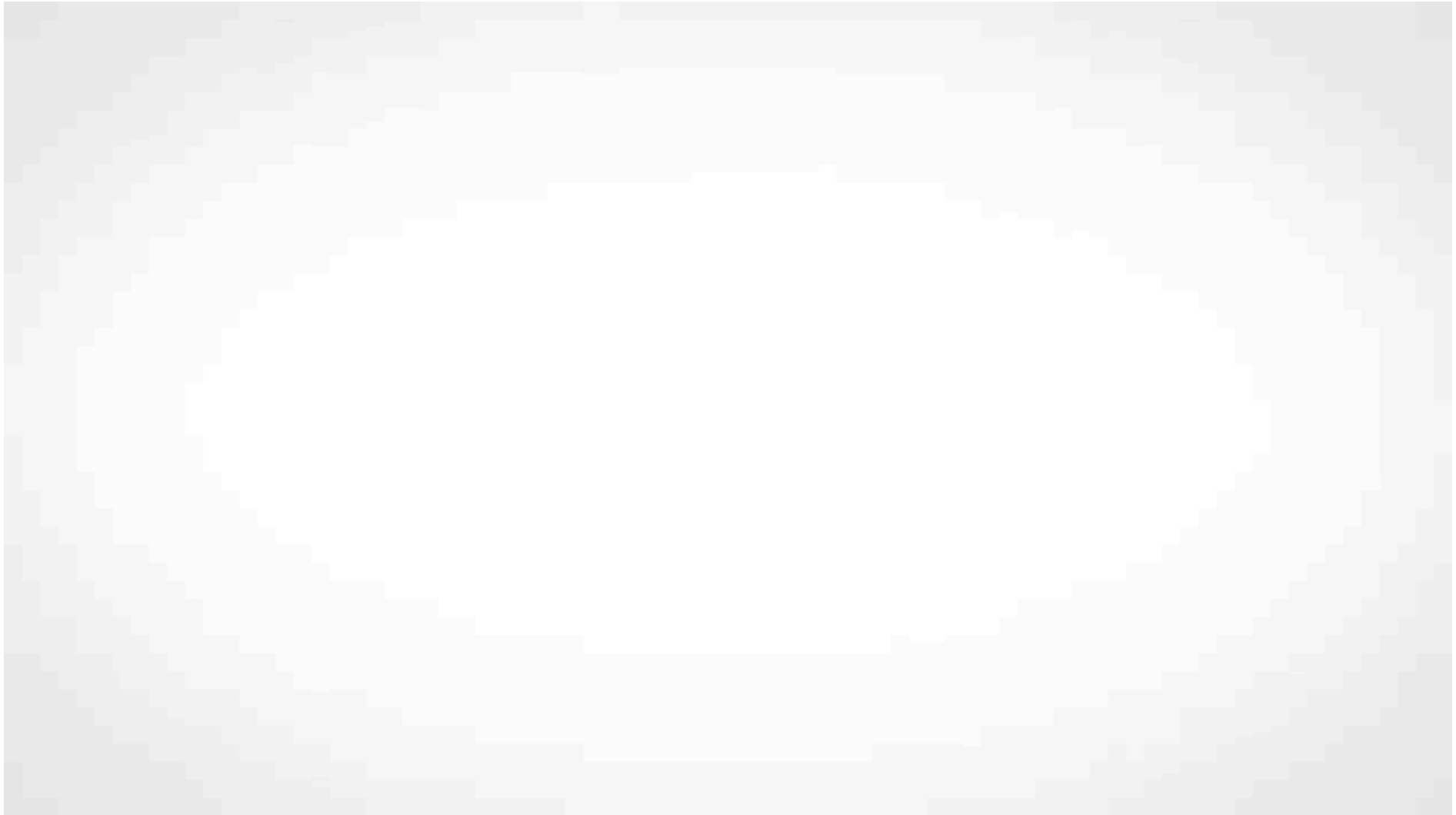
Source : [businesscomputingworld.co.uk](http://businesscomputingworld.co.uk)











**The concept of “Uptime” was pioneered by the Uptime Institute which was founded in 1993 and introduced its well defined Tier Classification system: I, II, III and IV, of which Tier IV represents the highest level of projected availability.**

**TIER 1 :**

Disponibility : 99.67%,  
no more 28.8 hours downtime/year  
No redundancy

**TIER II :**

Disponibility : 99.75%,  
no more 22 hours downtime/year  
Partial redundancy

**TIER III :**

Disponibility : 99.982 %,  
no more 1,5 hours downtime/year  
Redundancy N+1

**TIER IV :**

Disponibility : 99.995 %,  
No more 0,8 hours downtime/year  
Redundancy 2N+1

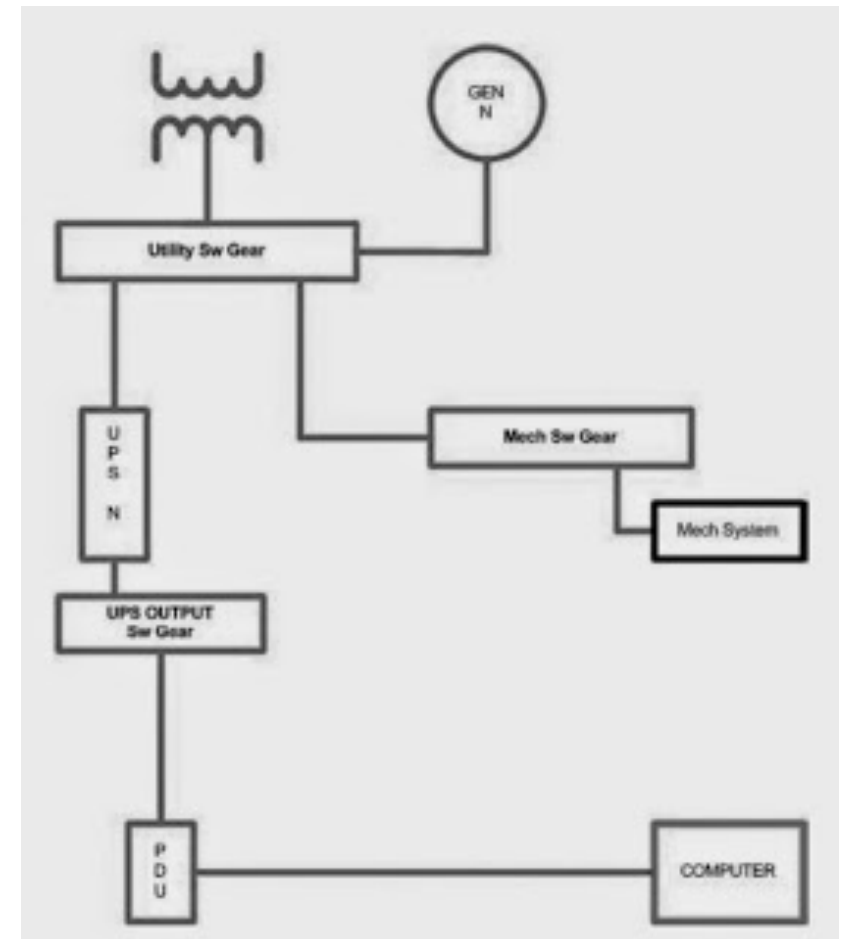
The terms “N, N+1 and 2N”, typically refer to the number of power and cooling components that comprise the entire data center infrastructure systems.

Tier Requirement	Tier 1	Tier II	Tier III	Tier IV
Source	System	System	System	System + System
System Component Redundancy	N	N+1	N+1	Minimum of N+1
Distribution Paths	1	1	1 normal and 1 alternate	2 simultaneously active
Compartmentalization	No	No	No	Yes
Concurrently Maintainable	No	No	Yes	Yes
Fault Tolerance (single event)	No	No	No	Yes

## TIER I

### TIER I :

- Components capacity to support IT load is 100%
- Single distribution path
- Rack power < 1kW Multi-points of failure and human errors
- Schedule maintenance twice per annum last for 12 hours
- Availability 99.67%, annual down time max. 28.8 hours

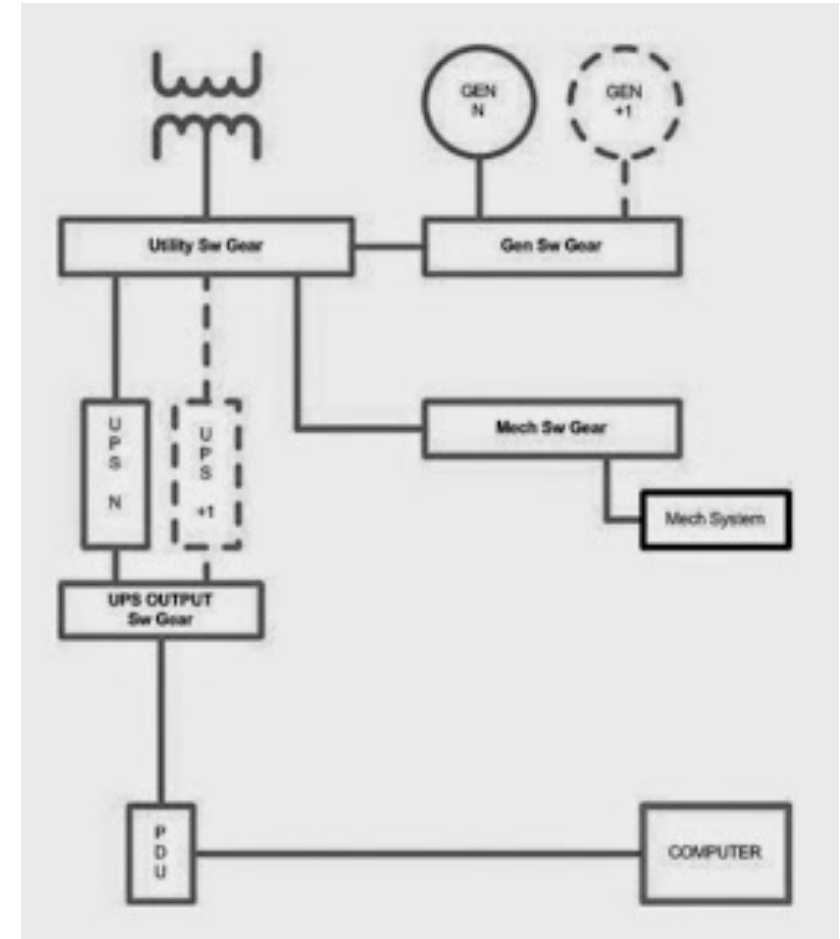




## TIER II

### TIER II :

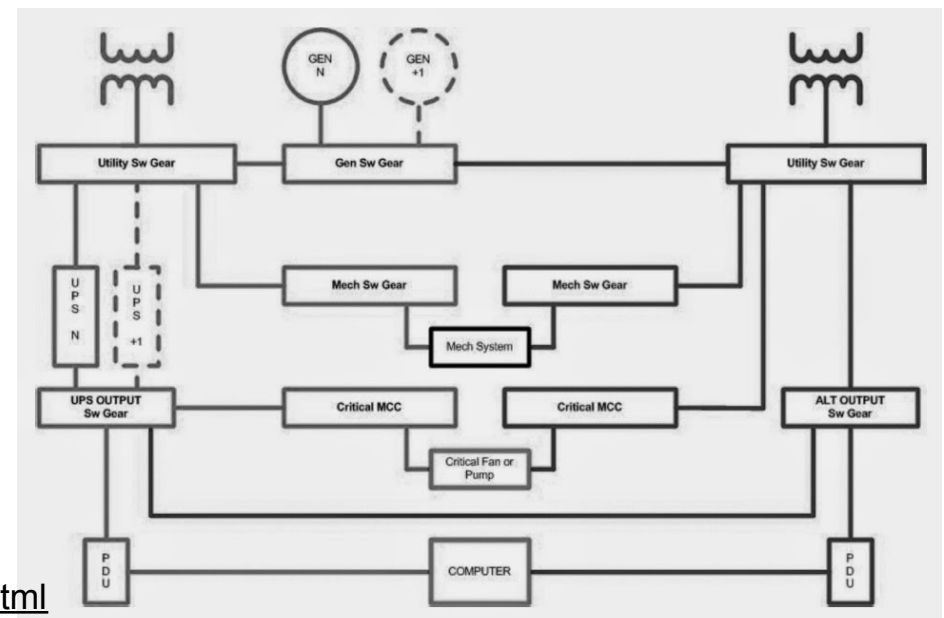
- Components capacity to support IT load is N+1
- Redundant for critical components
- Single distribution path
- Rack power < 2kW
- Multi-points of failure and human errors
- Schedule maintenance 3 times every 2 years each 12 hours
- Availability 99.75%, annual down time max. 22 hours



## TIER III

### TIER II :

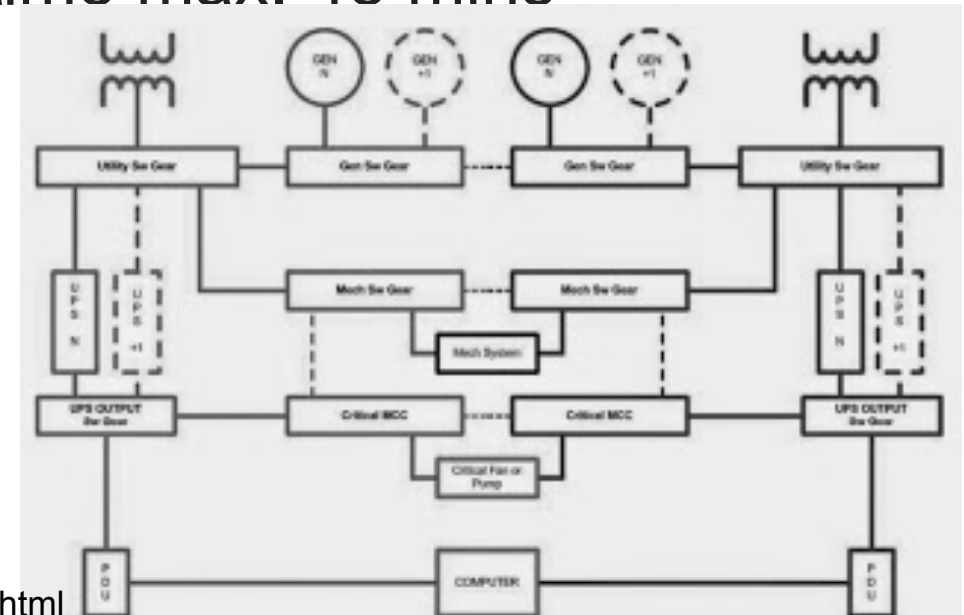
- Components capacity to support IT load is N+1
- Redundant for all components and distribution path
- Allow concurrent maintenance
- Rack power > 3kW
- Require MV supply (11kV in Hong Kong)
- Some failure points and human errors
- Schedule maintenance not required
- Availability 99.98%, annual down time max. 96 mins



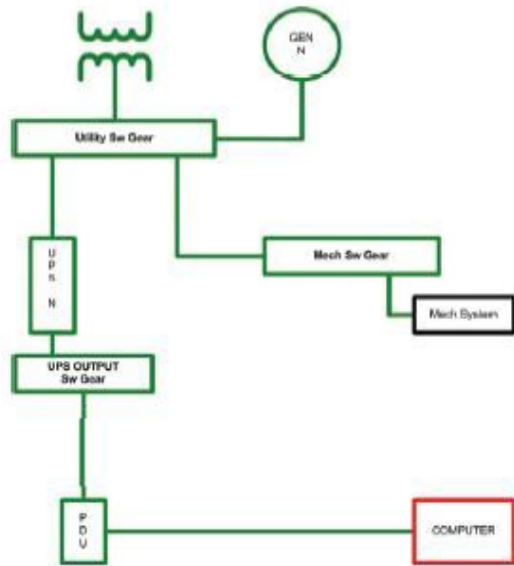
## TIER IV

### TIER IV :

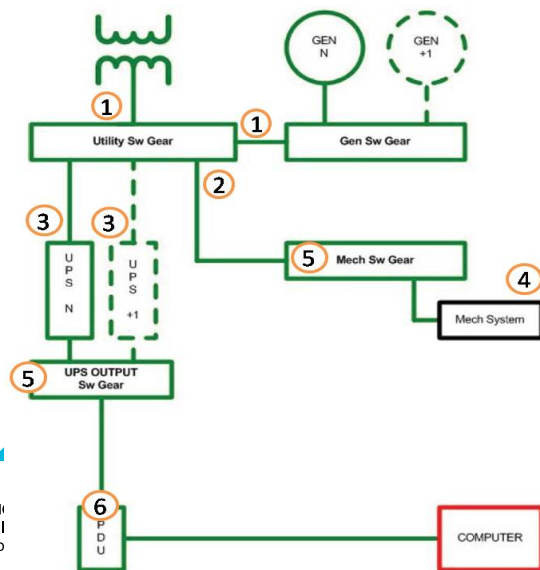
- Components capacity to support IT load is N after any failure
- Dual active distribution path
- No Single Point of Failure
- Rack power > 4kW
- Require MV supply (11kV in Hong Kong)
- Fault Tolerant, Fire, EPO and human errors
- Schedule maintenance not required
- Availability 99.99%, annual down time max. 48 mins



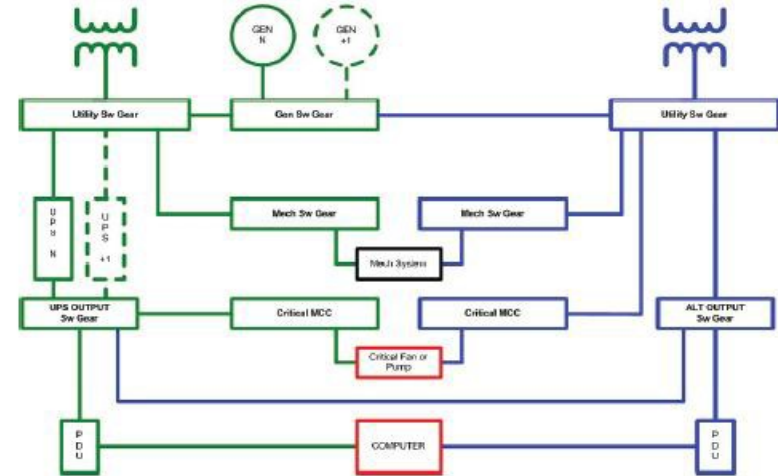
## Tier I



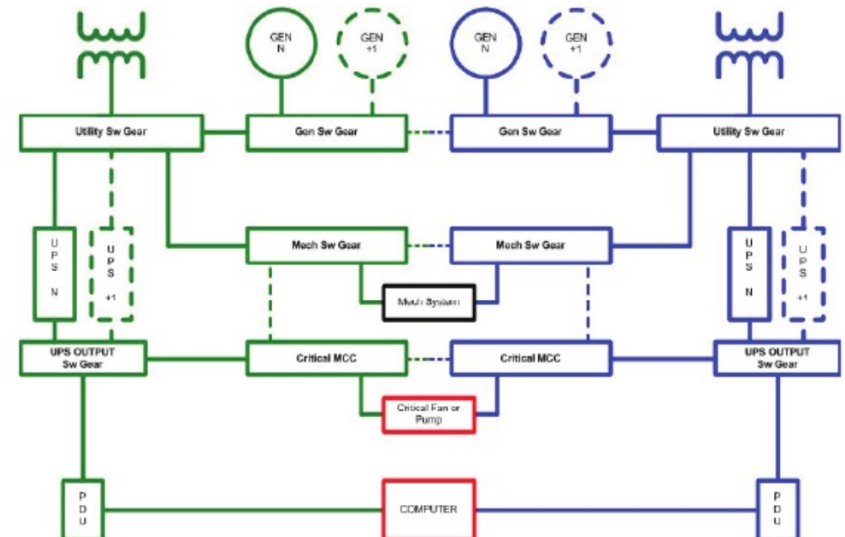
## Tier II



## Tier III



## Tier IV



An “N” system is not redundant at all, and the failure of any component will cause an outage, effectively describing a tier 1 type facility.

N+1 and 2N, represent increasing levels of component redundancies and power paths, roughly mapping to the tiers 2-4, however it is **important to understand that redundant components in themselves do not guarantee continuous availability.**

Besides redundancy, the ability to do planned maintenance or emergency repairs on systems may involve the necessity to take them offline

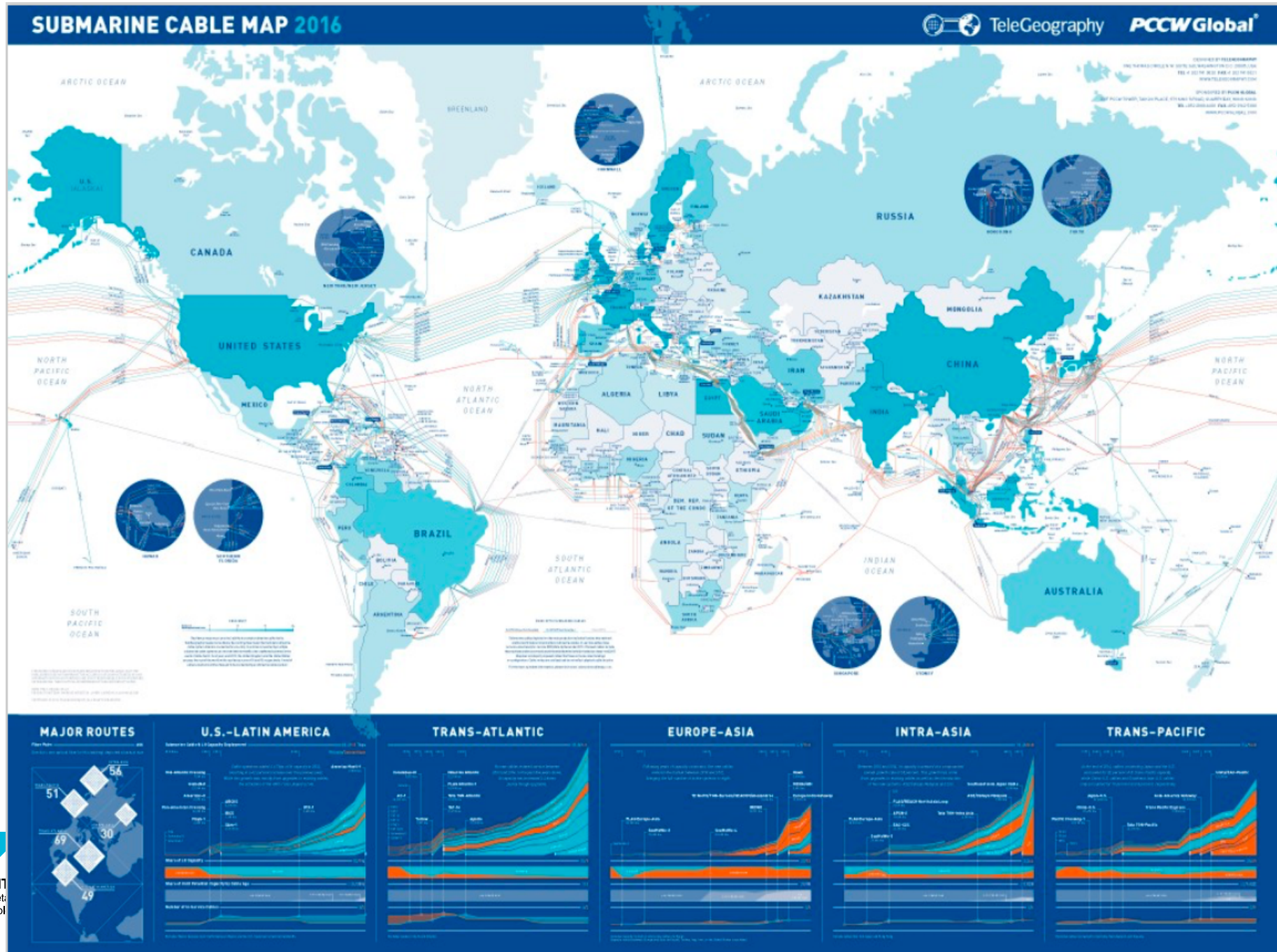






# WORLD NETWORK 2016

35



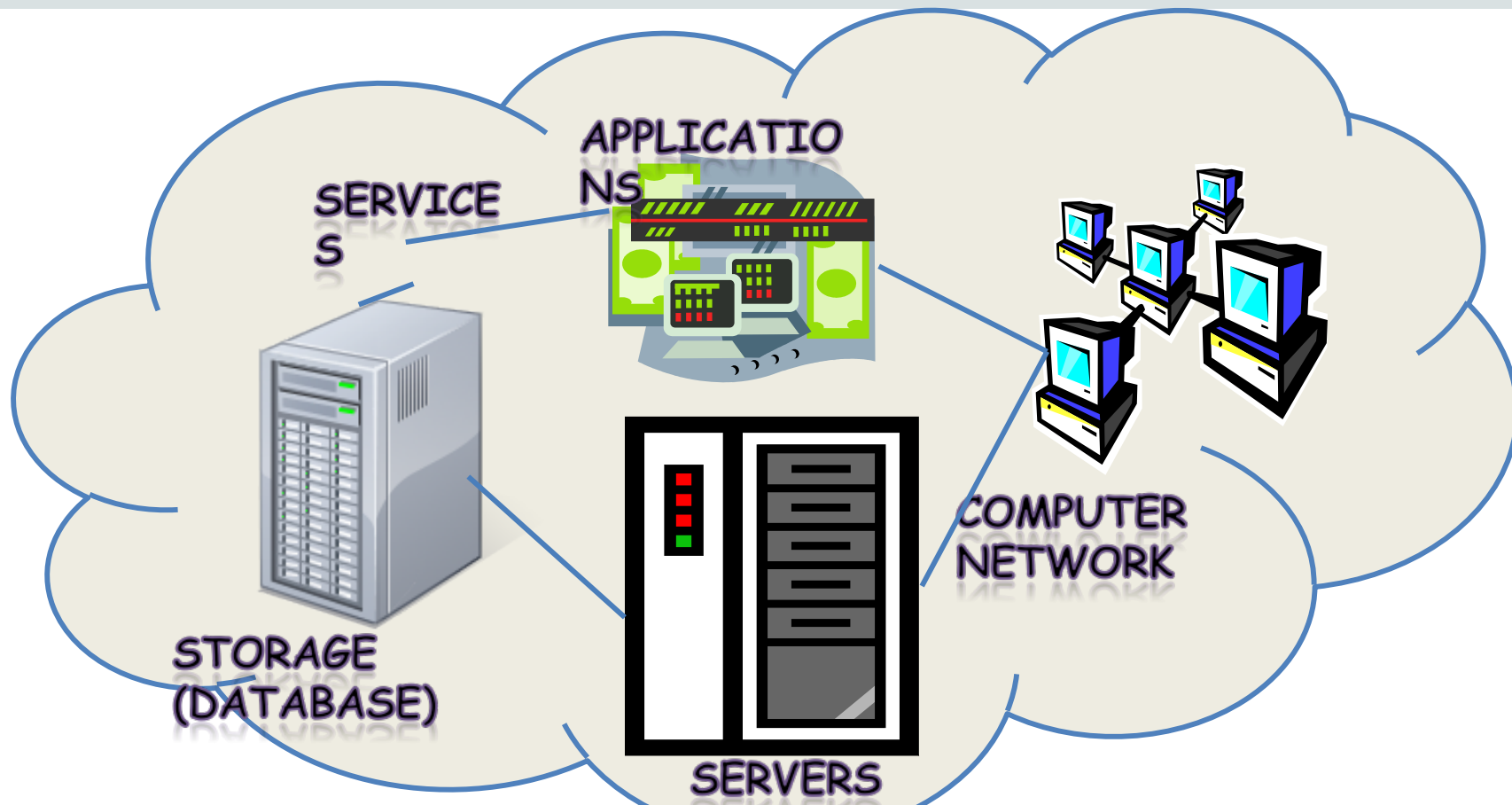
# CHAPTER 3

## INSIDE THE CLOUD MODEL



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom



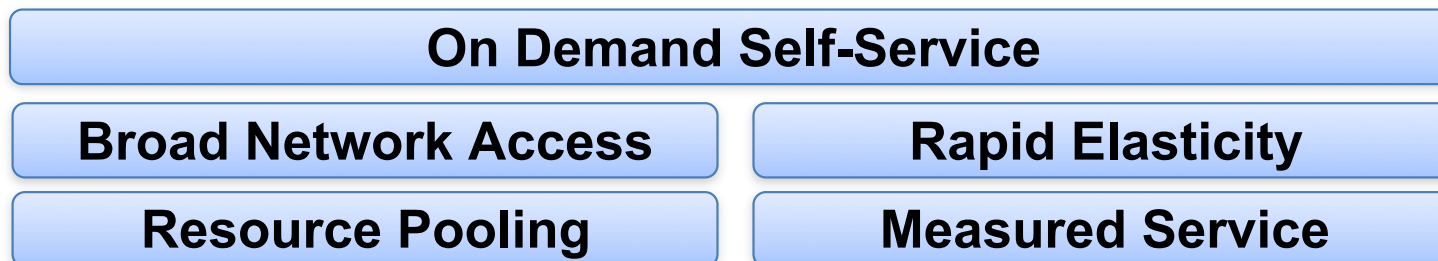


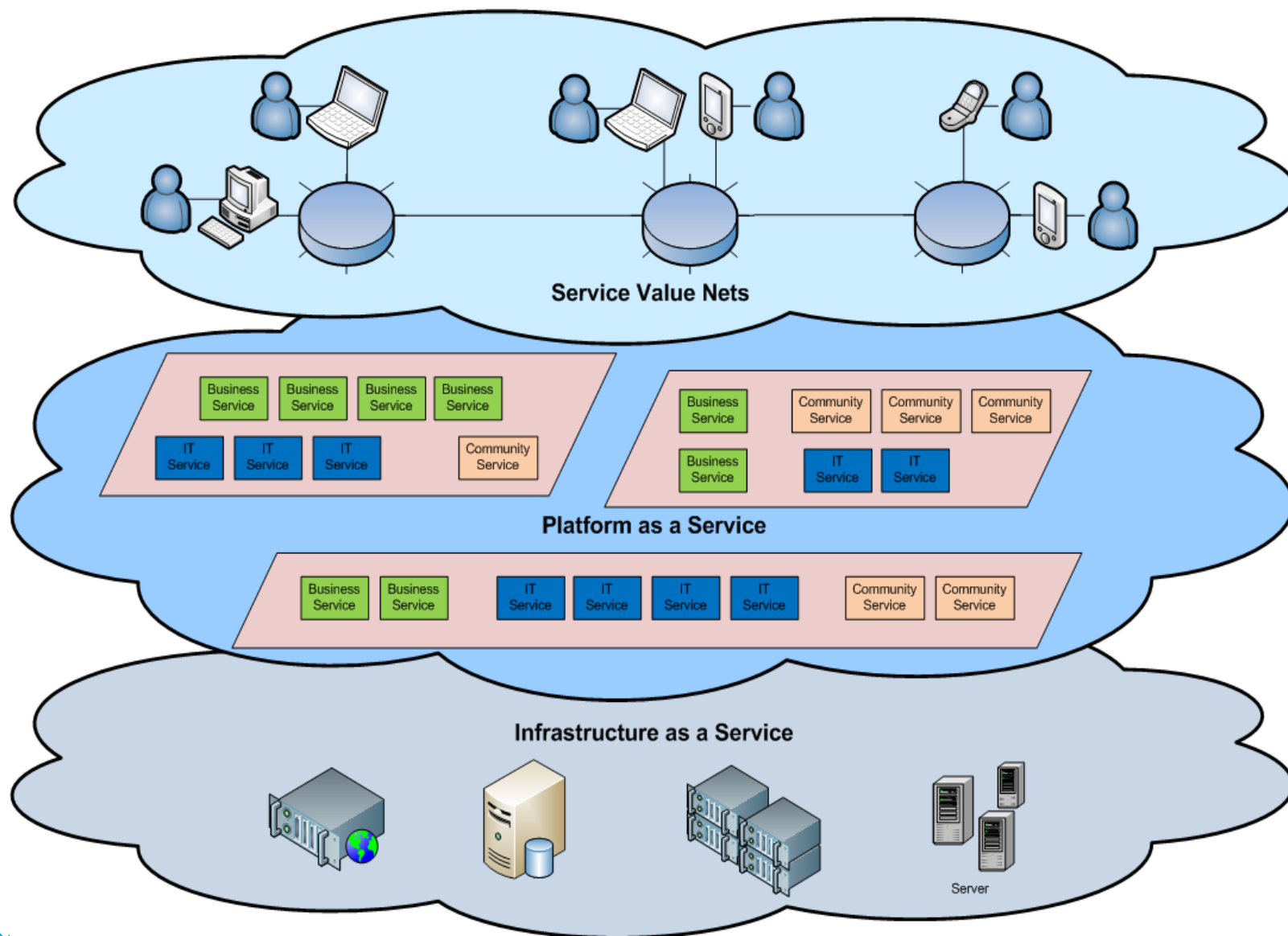
- Shared pool of configurable computing resources
- On-demand network access
- Provisioned by the Service Provider

## Common Characteristics:



## Essential Characteristics:





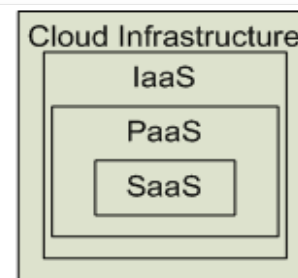
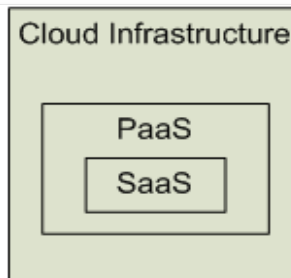
## Software as a Service (SaaS)

## Platform as a Service (PaaS)

## Infrastructure as a Service (IaaS)

SalesForce CRM

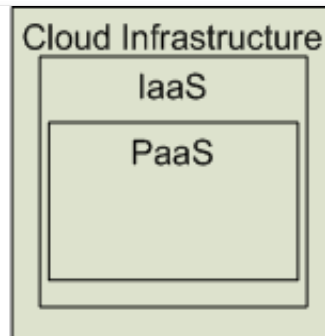
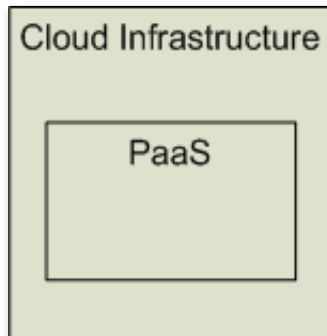
LotusLive



Software as a Service (SaaS)  
Providers  
Applications



Google App Engine

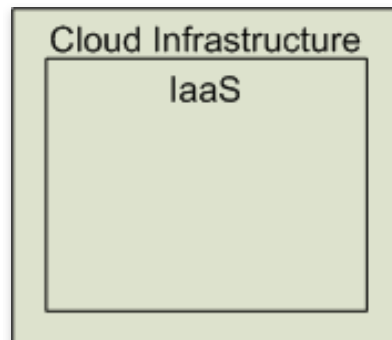


Platform as a Service (PaaS)

Deploy customer created Applications



IMT Atlantique  
Bretagne-Pays de la Loire  
Ecole Mines-Télécom



Infrastructure as a Service (IaaS)

Rent Processing, storage, N/W capacity & computing resources

<b>Application Service (SaaS)</b>	MS Live/ExchangeLabs, IBM, Google Apps; Salesforce.com Quicken Online, Zoho, Cisco
<b>Application Platform</b>	Google App Engine, Mosso, Force.com, Engine Yard, Facebook, Heroku, AWS
<b>Server Platform</b>	3Tera, EC2, SliceHost, GoGrid, RightScale, Linode
<b>Storage Platform</b>	Amazon S3, Dell, Apple, ...



		Services	Description
Application Focused		Services	Services - Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa
		Application	Application - Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online
		Development	Development - Software development platforms used to build custom cloud based applications (PAAS & SAAS) such as Salesforce
Infrastructure Focused		Platform	Platform - Cloud based platforms, typically provided using virtualization, such as Amazon ECC, Sun Grid
		Storage	Storage - Data storage or cloud based NAS such as CTERA, iDisk, CloudNAS
		Hosting	Hosting - Physical data centers such as those run by IBM, HP, NaviSite, etc.

# CHAPTER 4

## IAS

# INFRASTRUCTURE AS A SERVICE



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

# CHAPTER 4.1

## VIRTUALIZATION



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

## Aggregation

- Make  $N$  resources appear as 1 (clusters / grids)

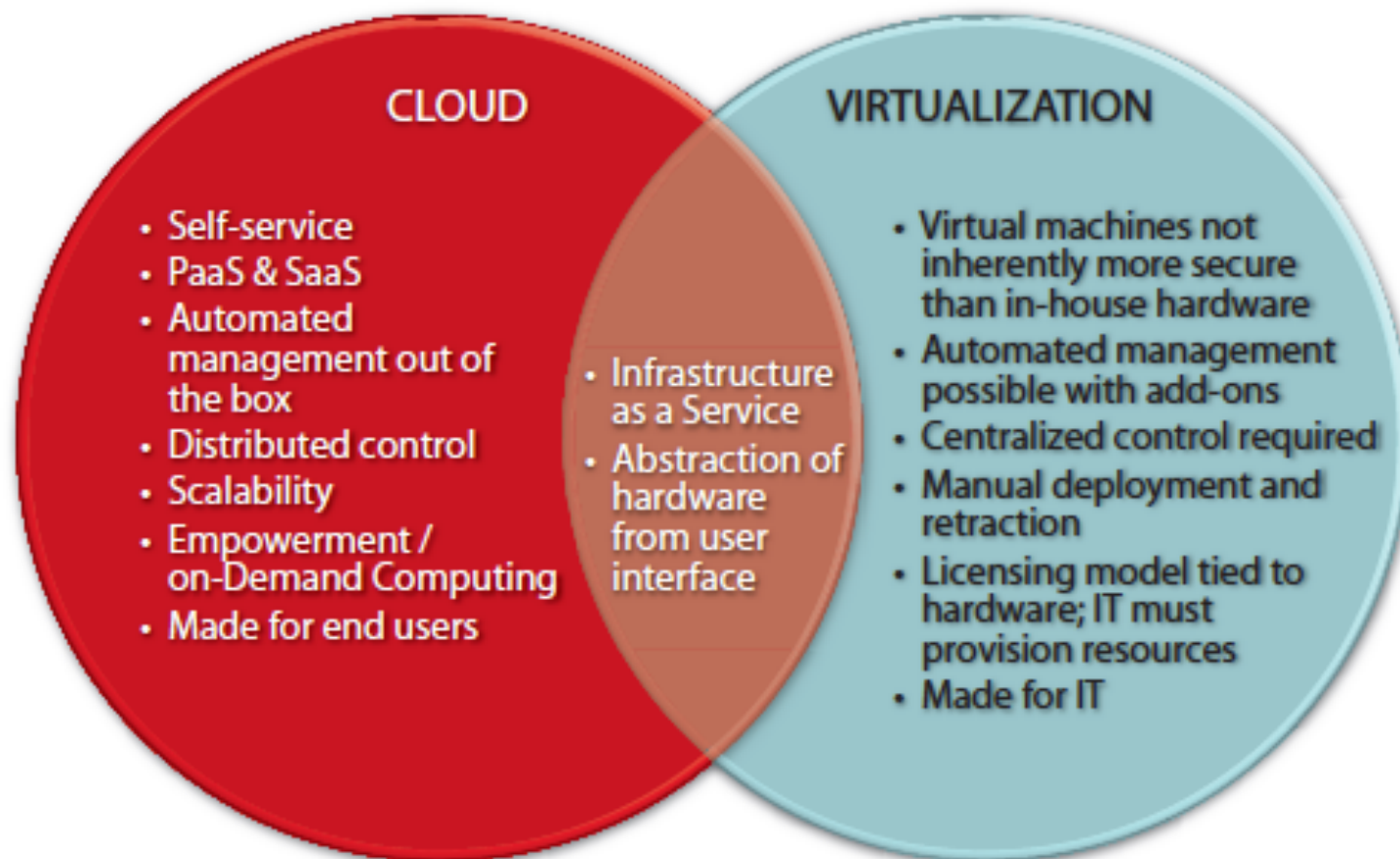
## Partition / Replication:

- Make a resource appear as  $N$

## Translation (Emulation)

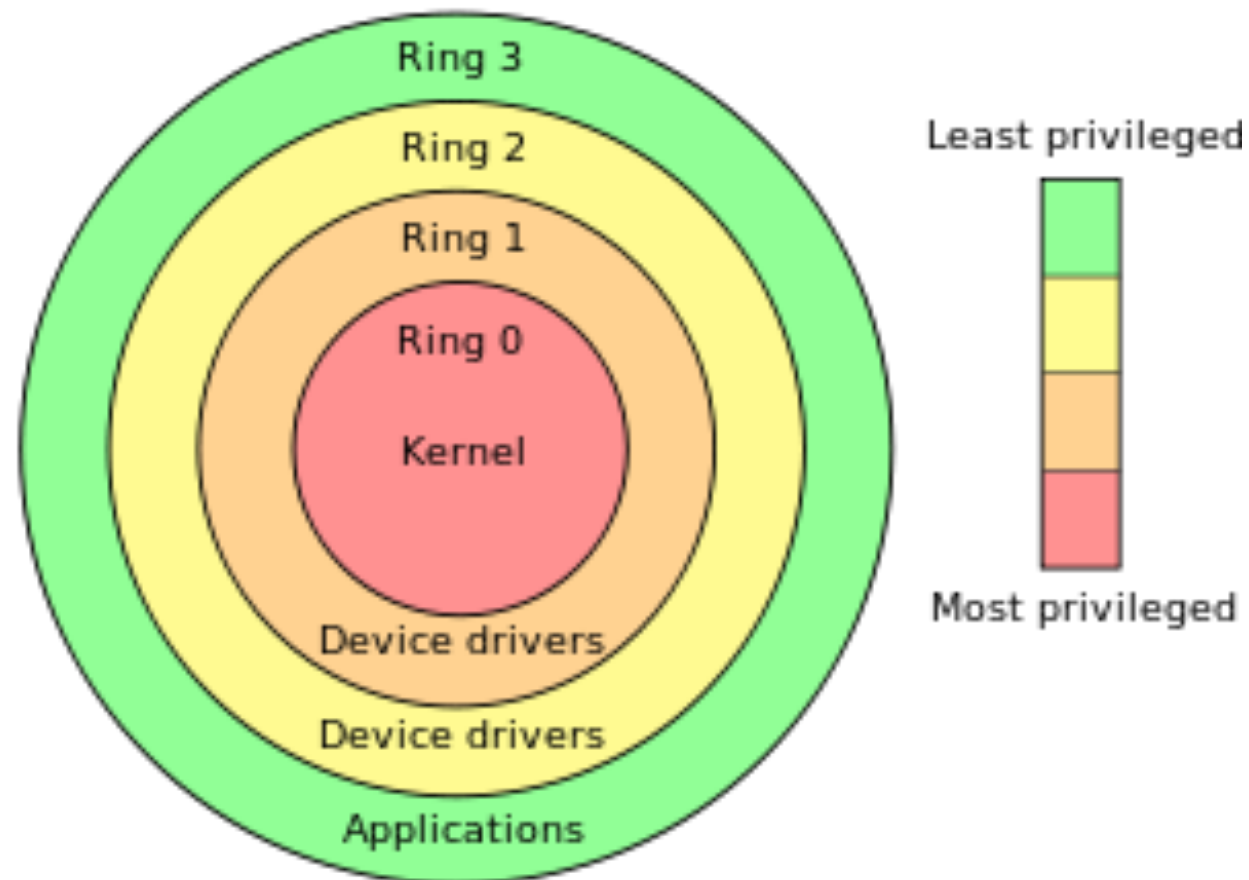
- Make  $X$  appear as  $Y$  (sometimes  $X$  is identical to  $Y$ )
- May be combined with “partition”

Mostly interested in “partition”

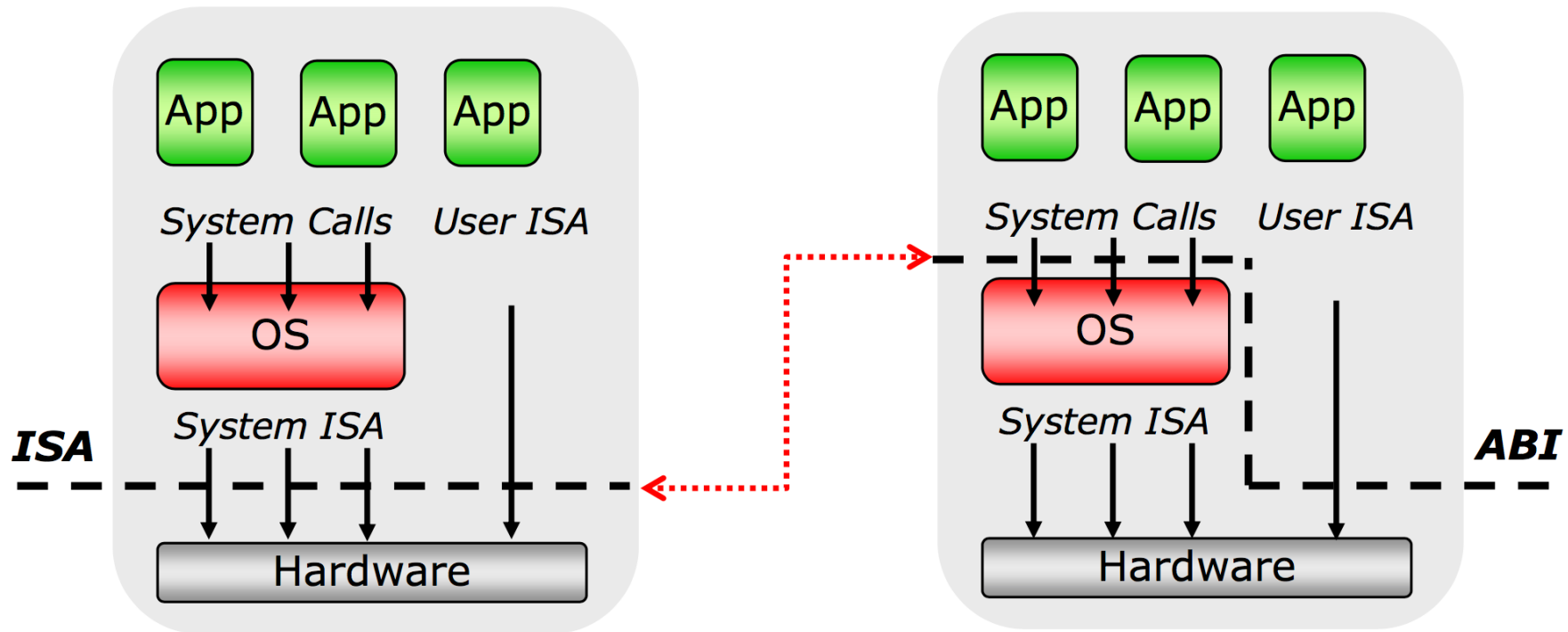




- First appeared in IBM mainframes in 1972
- Allowed multiple users to share a batch-oriented system
- Formal definition of virtualization helped move it beyond IBM
  - A VMM provides an environment for programs that is essentially identical to the original machine
  - Programs running within that environment show only minor performance decreases
  - The VMM is in complete control of system resources
- In late 1990s Intel CPUs fast enough for researchers to try virtualizing on general purpose PCs
  - Xen and VMware created technologies, still used today
  - Virtualization has expanded to many OSES, CPUs, VMMs

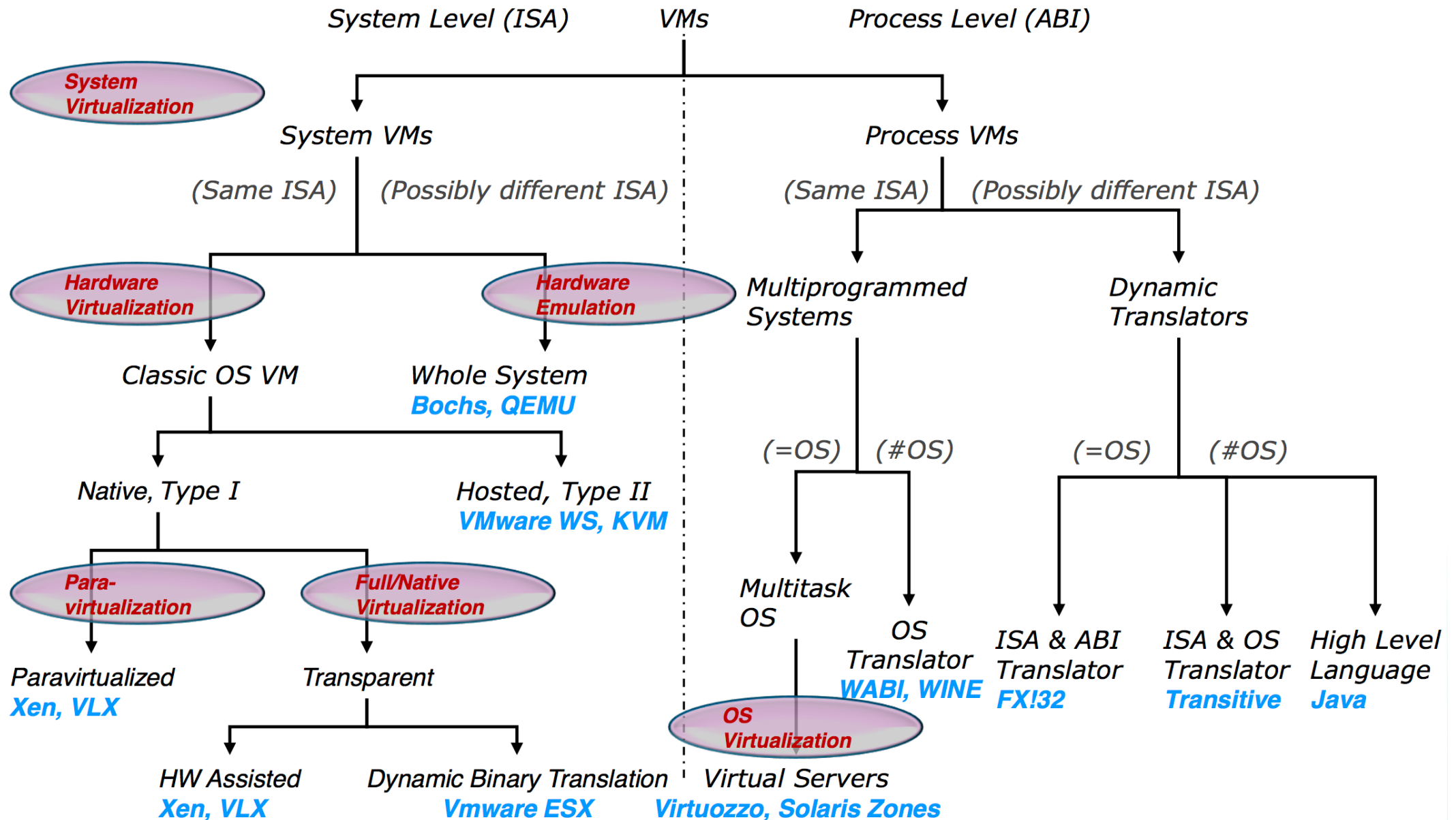


- Systems built of : hardware, OS, applications
- 2 main interfaces: ISA (hardware), ABI (OS)



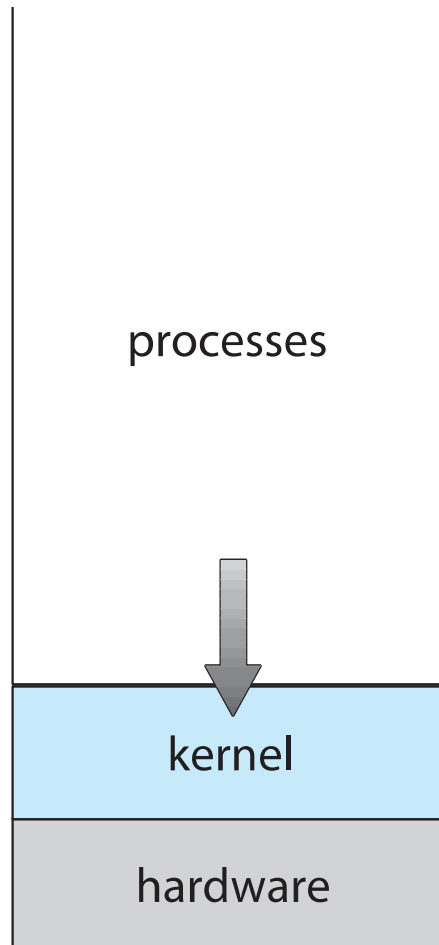
System level VM provide an ISA interface

Process level VM provide an ABI interface

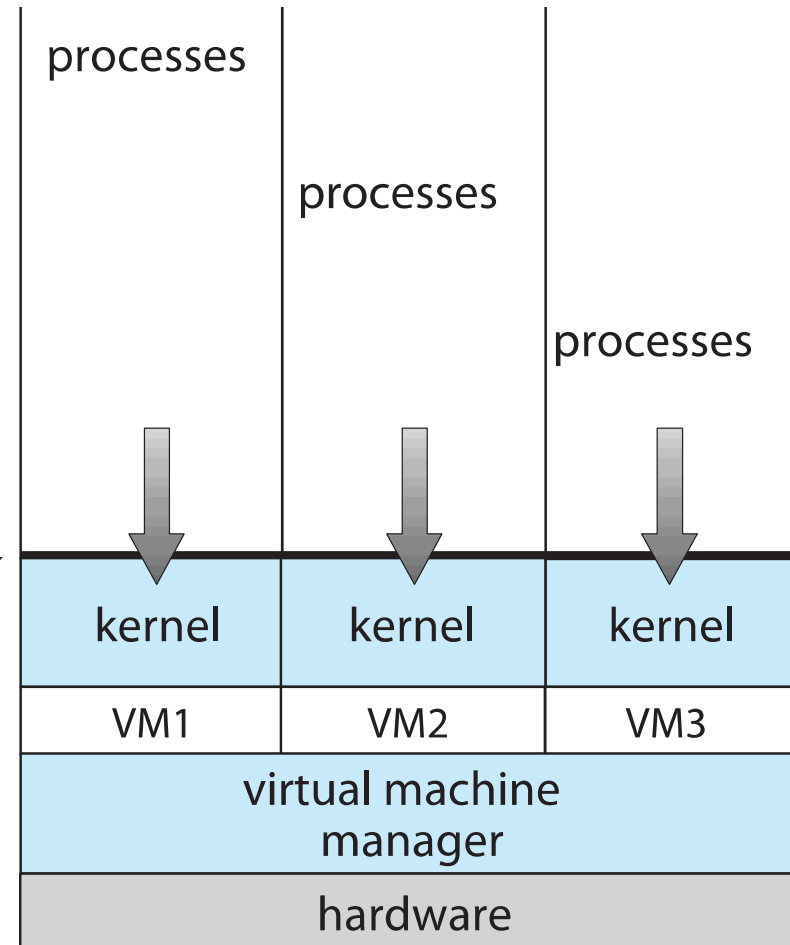


- Fundamental idea – abstract hardware of a single computer into several different execution environments
  - Similar to layered approach
  - But layer creates virtual system (**virtual machine**, or **VM**) on which operation systems or applications can run
- Several components
  - **Host** – underlying hardware system
  - **Virtual machine manager (VMM)** or **hypervisor** – creates and runs virtual machines by providing interface that is **identical** to the host
    - ▶ (Except in the case of paravirtualization)
  - **Guest** – process provided with virtual copy of the host
    - ▶ Usually an operating system
- Single physical machine can run multiple operating systems concurrently, each in its own virtual machine





programming interface



(a) Nonvirtual machine

(b) Virtual machine

- Vary greatly, with options including:
  - **Type 0 hypervisors** - Hardware-based solutions that provide support for virtual machine creation and management via firmware
    - ▶ IBM LPARs and Oracle LDOMs are examples
  - **Type 1 hypervisors** - Operating-system-like software built to provide virtualization
    - ▶ Including VMware ESX, Joyent SmartOS, and Citrix XenServer
  - **Type 1 hypervisors** – Also includes general-purpose operating systems that provide standard functions as well as VMM functions
    - ▶ Including Microsoft Windows Server with HyperV and RedHat Linux with KVM
  - **Type 2 hypervisors** - Applications that run on standard operating systems but provide VMM features to guest operating systems
    - ▶ Including VMware Workstation and Fusion, Parallels Desktop, and Oracle VirtualBox

## Issues:

- Run multiple OS's
- OS designed assuming it is the only software controlling the physical resources of the machine.
- Need to detect and resolve conflicts such as masking interrupts, initiating an I/O, or programming the MMU by providing the expected behavior within the Virtual Machine, not necessarily on the physical level.
- In brief: detect sensitive instructions and fake them.
  - User mode to trap such instruction upon execution
  - Modify OS ahead of time
  - Hardware support

## ● Goal:

- Run the binary guest OS in user mode though it's been written to run in supervisor mode.

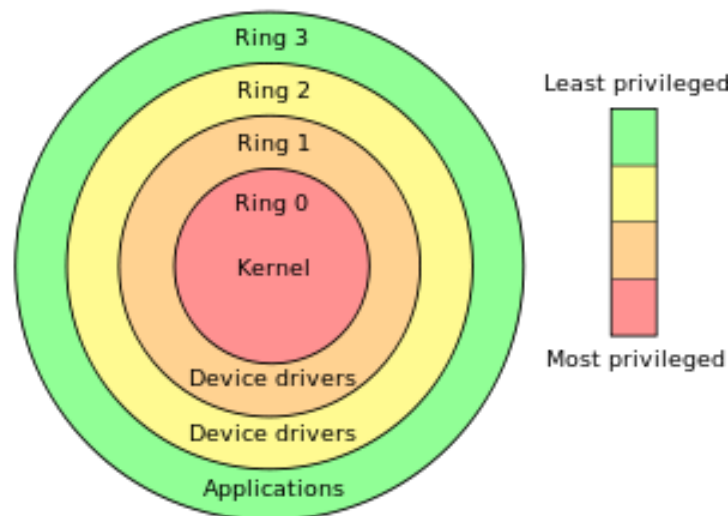
## ● Means:

- Transparent Virtualization: (full or native)
  - No modification of the OS image
  - Fully Virtualizable Processors (VT-x, AMD-V, IBM PPC)
  - Dynamic Binary Translation (VMware)
- Para-virtualization:
  - Modification of some of the OS HAL source files
  - (can be seen as a port to a new processor very similar to the real one).

- Other variations include:
  - **Paravirtualization** - Technique in which the guest operating system is modified to work in cooperation with the VMM to optimize performance
  - **Programming-environment virtualization** - VMMS do not virtualize real hardware but instead create an optimized virtual system
    - ▶ Used by Oracle Java and Microsoft.Net
  - **Emulators** – Allow applications written for one hardware environment to run on a very different hardware environment, such as a different type of CPU
  - **Application containment** - Not virtualization at all but rather provides virtualization-like features by segregating applications from the operating system, making them more secure, manageable
    - ▶ Including Oracle Solaris Zones, BSD Jails, and IBM AIX WPARs
- Much variation due to breadth, depth and importance of virtualization in modern computing

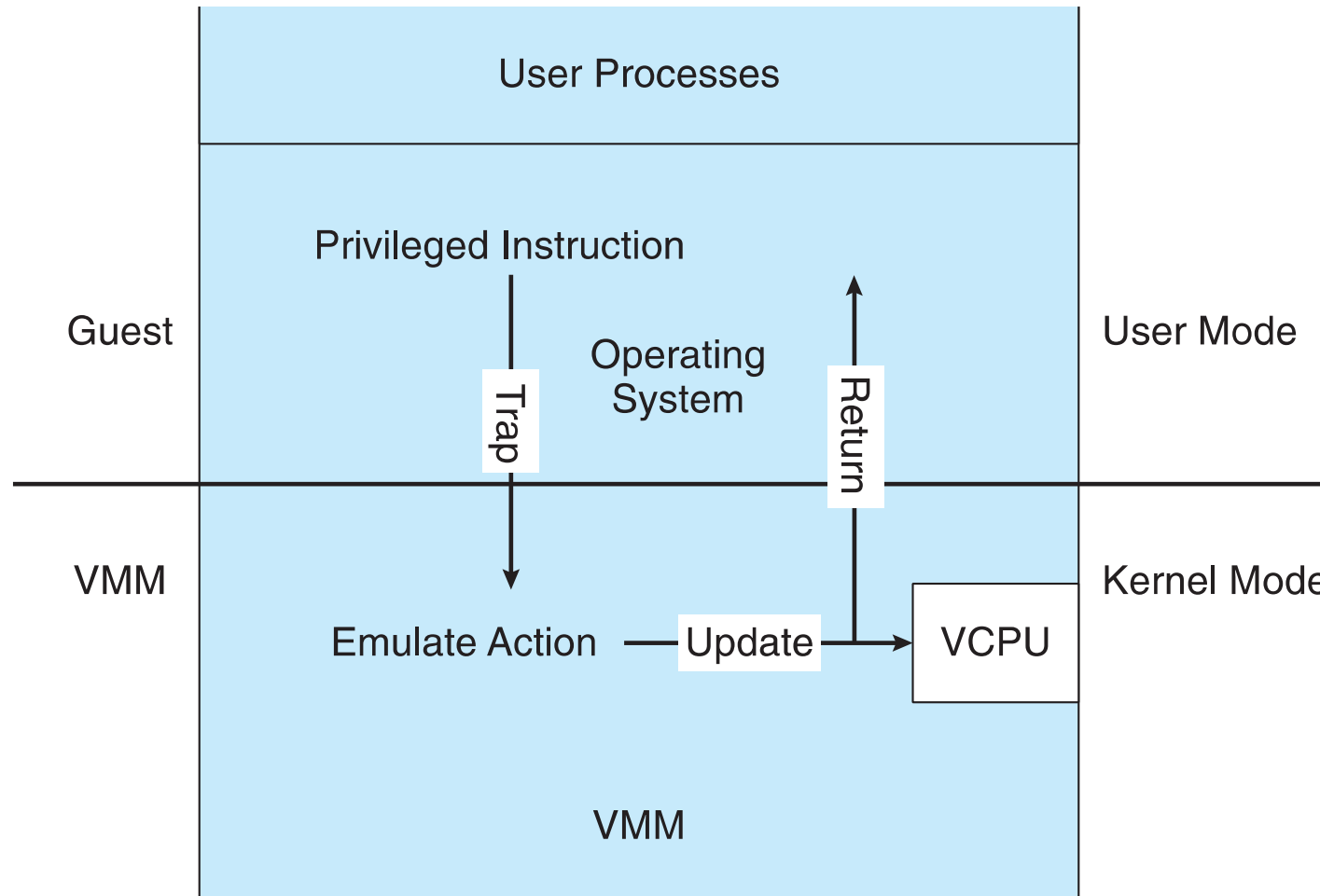


- Generally difficult to provide an **exact** duplicate of underlying machine
  - Especially if only dual-mode operation available on CPU
  - But getting easier over time as CPU features and support for VMM improves
  - Most VMMs implement **virtual CPU (VCPU)** to represent state of CPU per guest as guest believes it to be
    - ▶ When guest context switched onto CPU by VMM, information from VCPU loaded and stored
  - Several techniques, as described in next slides



- Dual mode CPU means guest executes in user mode
  - Kernel runs in kernel mode
  - Not safe to let guest kernel run in kernel mode too
  - So VM needs two modes – virtual user mode and virtual kernel mode
    - ▶ Both of which run in real user mode
  - Actions in guest that usually cause switch to kernel mode must cause switch to virtual kernel mode

- How does switch from virtual user mode to virtual kernel mode occur?
  - Attempting a privileged instruction in user mode causes an error -> trap
  - VMM gains control, analyzes error, executes operation as attempted by guest
  - Returns control to guest in user mode
  - Known as **trap-and-emulate**
  - Most virtualization products use this at least in part
- Trap to supervisor mode when executed from user mode
  - cli, sti (Intel x86) trap when executed from user-mode control to guest in user mode

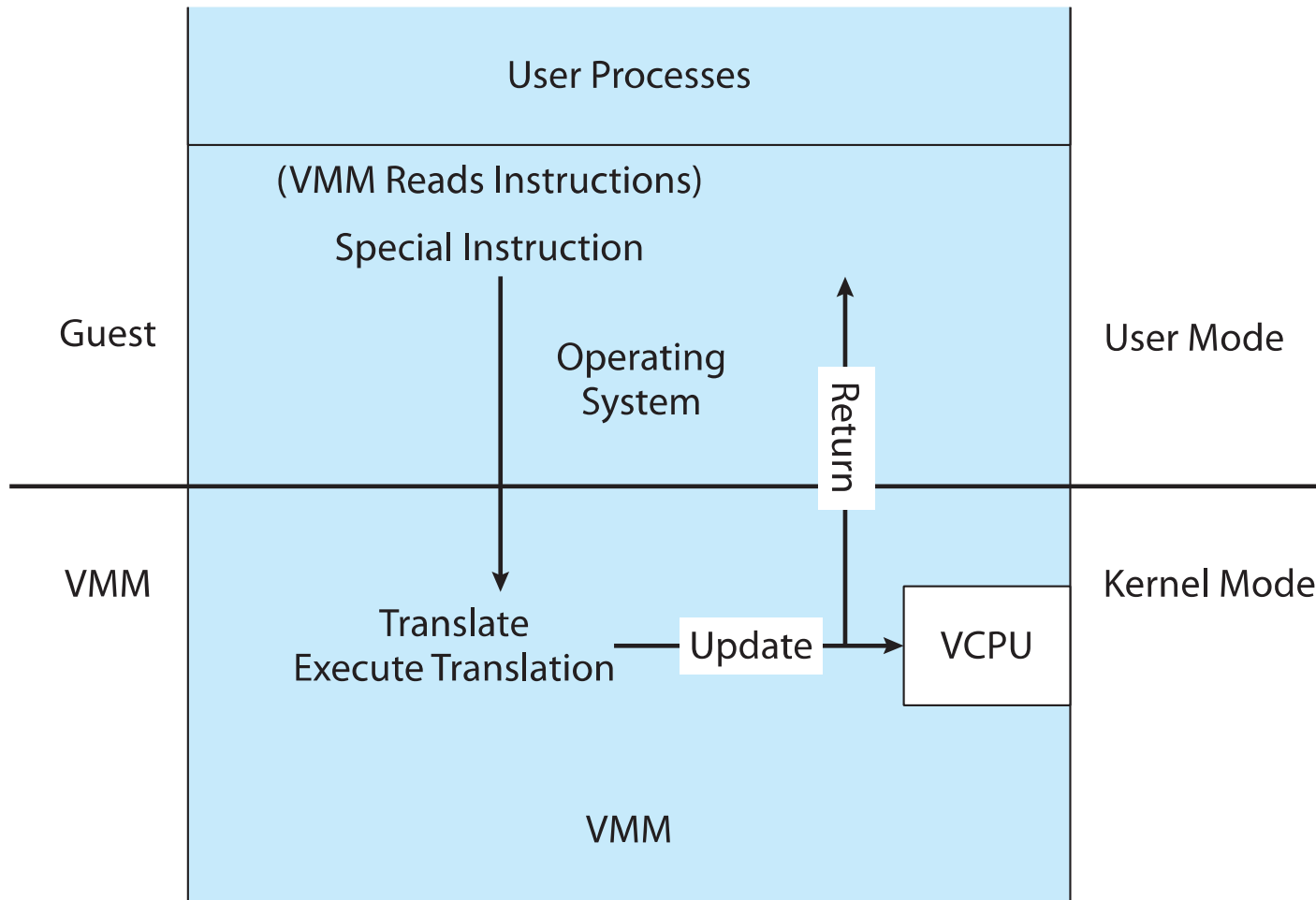


- User mode code in guest runs at same speed as if not a guest
- But kernel mode privilege mode code runs slower due to trap-and-emulate
  - Especially a problem when multiple guests running, each needing trap-and-emulate
- CPUs adding hardware support, mode CPU modes to improve virtualization performance

- Some CPUs don't have clean separation between privileged and nonprivileged instructions
  - Earlier Intel x86 CPUs are among them
    - ▶ Earliest Intel CPU designed for a calculator
  - Backward compatibility means difficult to improve
  - Consider Intel x86 **popf** instruction
    - ▶ Loads CPU flags register from contents of the stack
    - ▶ If CPU in privileged mode -> all flags replaced
    - ▶ If CPU in user mode -> on some flags replaced
      - No trap is generated



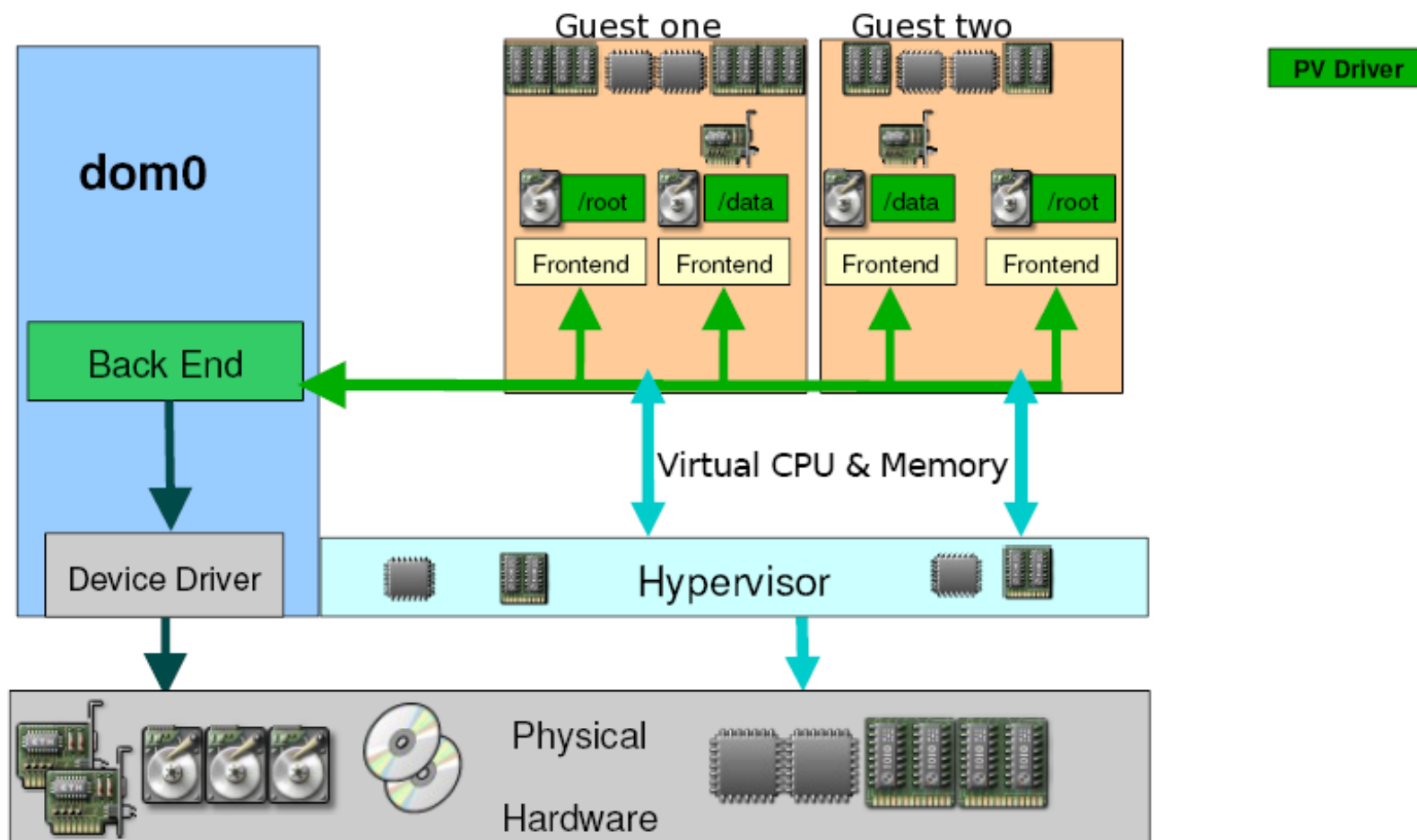
- Other similar problem instructions we will call ***special instructions***
  - Caused trap-and-emulate method considered impossible until 1998
- Binary translation solves the problem
  - Basics are simple, but implementation very complex
    1. If guest VCPU is in user mode, guest can run instructions natively
    2. If guest VCPU in kernel mode (guest believes it is in kernel mode)
      1. VMM examines every instruction guest is about to execute by reading a few instructions ahead of program counter
      2. Non-special-instructions run natively
      3. Special instructions translated into new set of instructions that perform equivalent task (for example changing the flags in the VCPU)

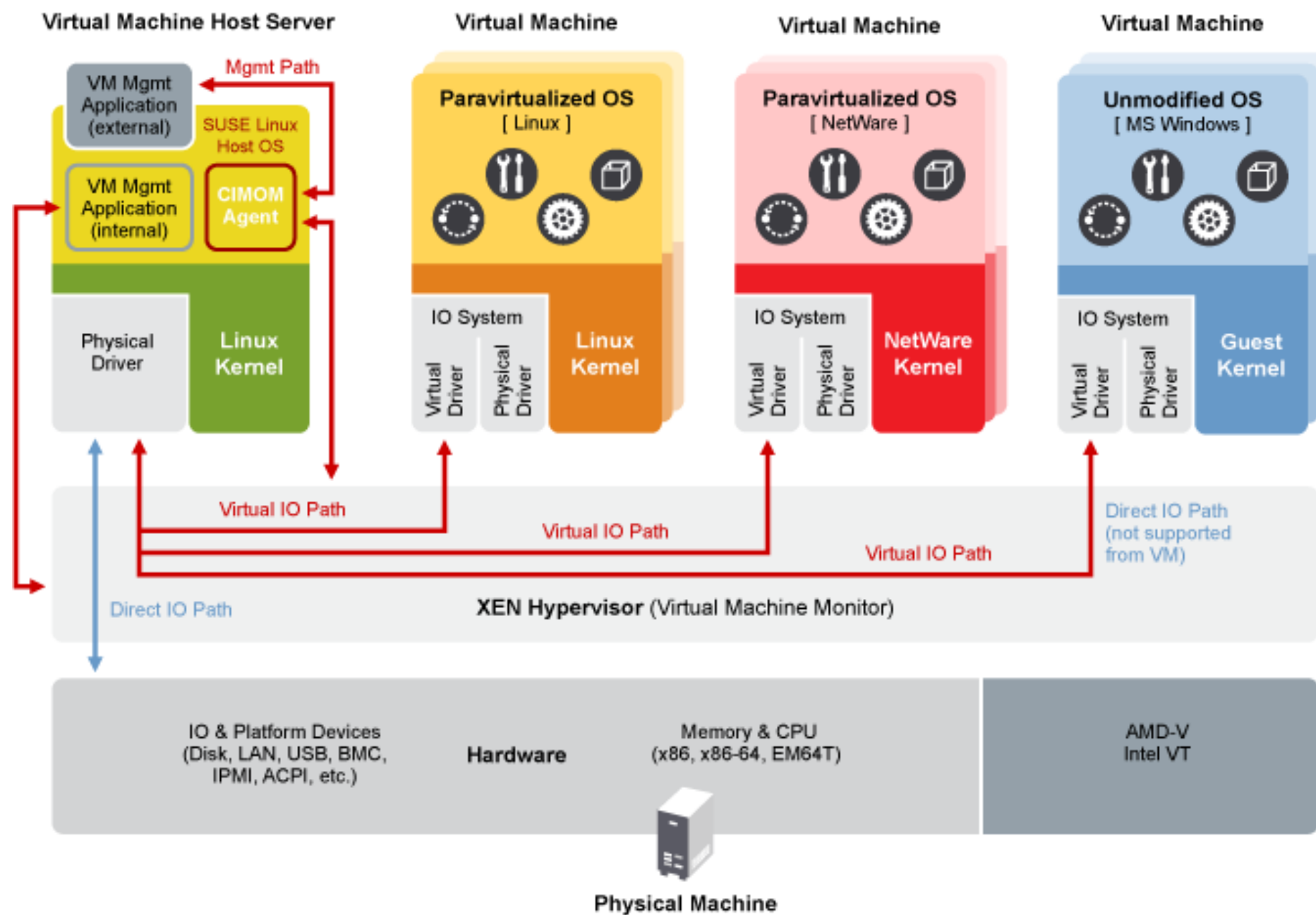


- Implemented by translation of code within VMM
- Code reads native instructions dynamically from guest, on demand, generates native binary code that executes in place of original code
- Performance of this method would be poor without optimizations
  - Products like VMware use caching
    - ▶ Translate once, and when guest executes code containing special instruction cached translation used instead of translating again
    - ▶ Testing showed booting Windows XP as guest caused 950,000 translations, at 3 microseconds each, or 3 second (5 %) slowdown over native

- Does not *really* fit the definition of virtualization – VMM not presenting an exact duplication of underlying hardware
  - But still useful!
  - VMM provides services that guest must be modified to use
  - Leads to increased performance
  - Less needed as hardware support for VMs grows
- Xen, leader in paravirtualized space, adds several techniques
  - For example, clean and simple device abstractions
    - ▶ Efficient I/O
    - ▶ Good communication between guest and VMM about device I/O
    - ▶ Each device has circular buffer shared by guest and VMM via shared memory

# Xen Para-virtualization Architecture





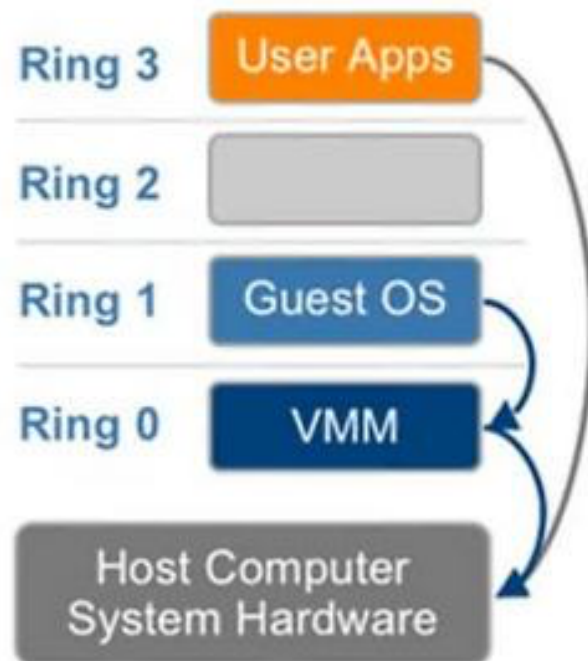


- Guest uses hypercall (call to hypervisor)
- Paravirtualization allowed virtualization of older x86 CPUs (and others) without binary translation
- Guest had to be modified to use run on paravirtualized VMM
- But on modern CPUs Xen no longer requires guest modification -> no longer paravirtualization

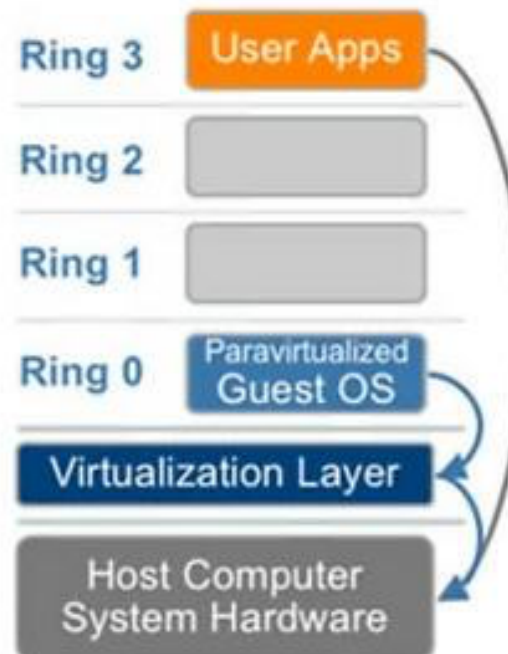
- All virtualization needs some HW support
- More support -> more feature rich, stable, better performance of guests
- Intel added new **VT-x** instructions in 2005 and AMD the **AMD-V** instructions in 2006
  - CPUs with these instructions remove need for binary translation
  - Generally define more CPU modes – “guest” and “host”
  - VMM can enable host mode, define characteristics of each guest VM, switch to guest mode and guest(s) on CPU(s)
  - In guest mode, guest OS thinks it is running natively, sees devices (as defined by VMM for that guest)
    - ▶ Access to virtualized device, priv instructions cause trap to VMM
    - ▶ CPU maintains VCPU, context switches it as needed
- HW support for Nested Page Tables, DMA, interrupts as well over time

# Architectural Comparison

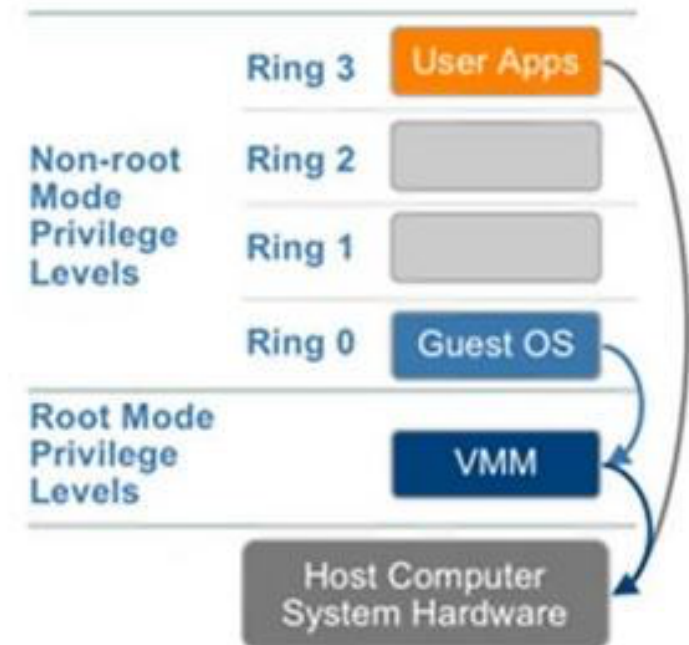
## Full Virtualization

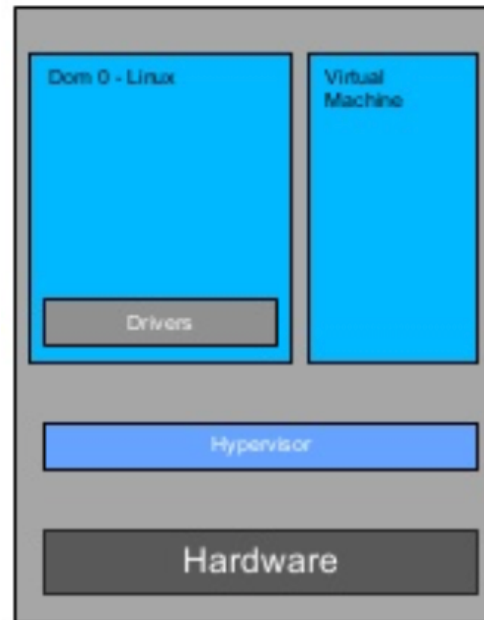


## Paravirtualization

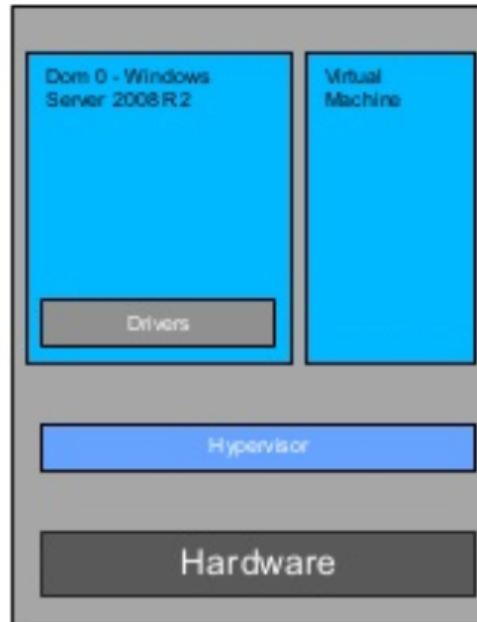


## Hardware Assisted

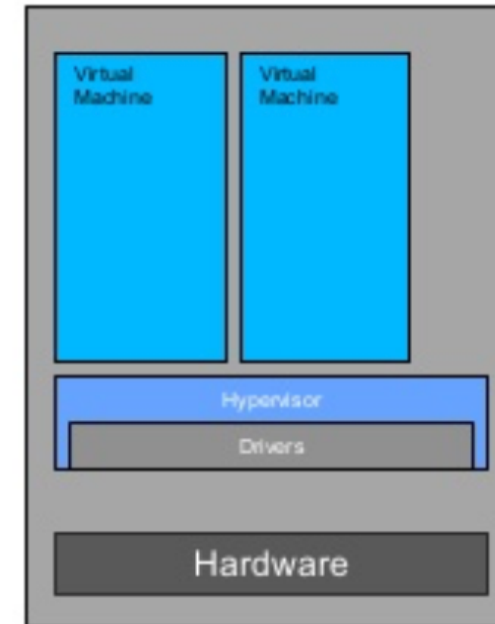




XenServer



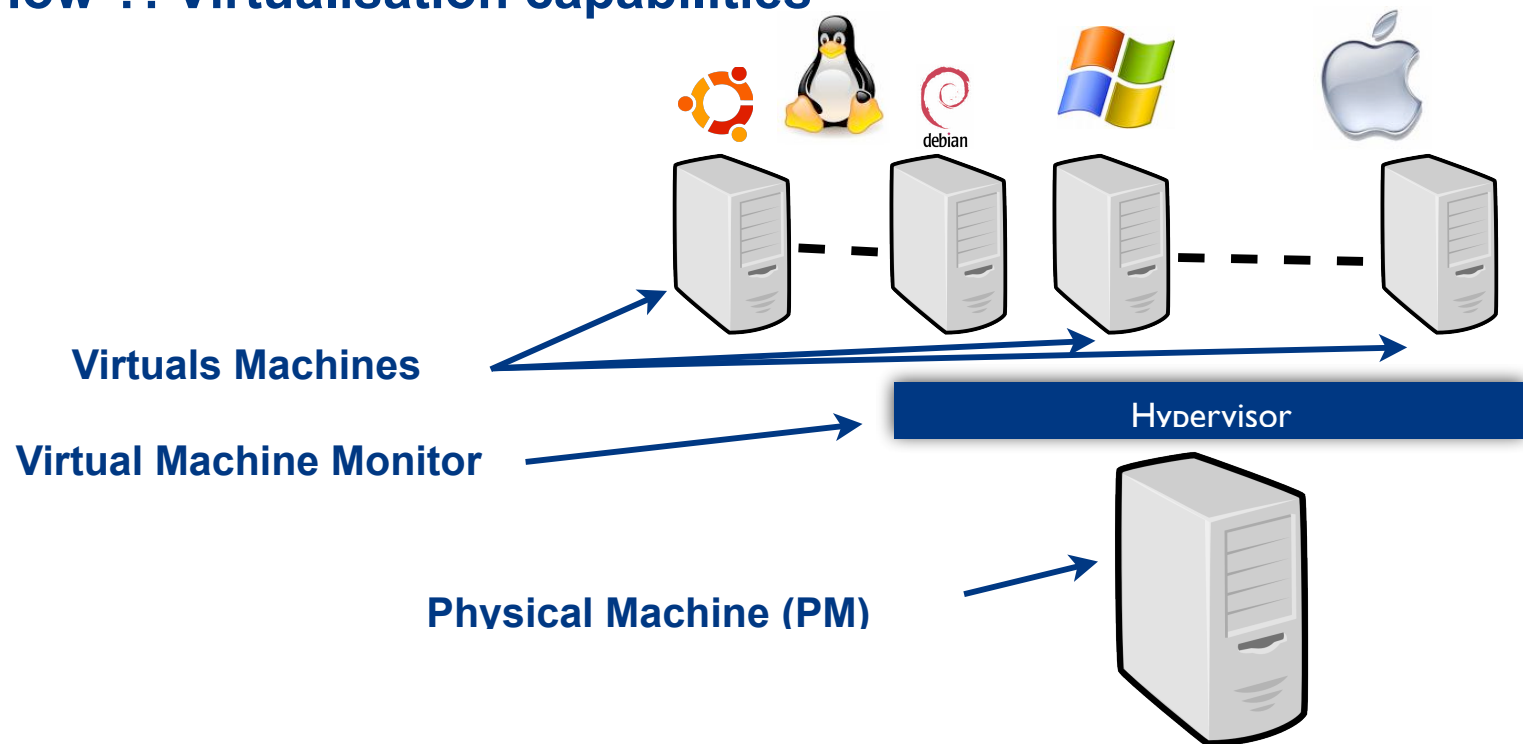
Hyper-V



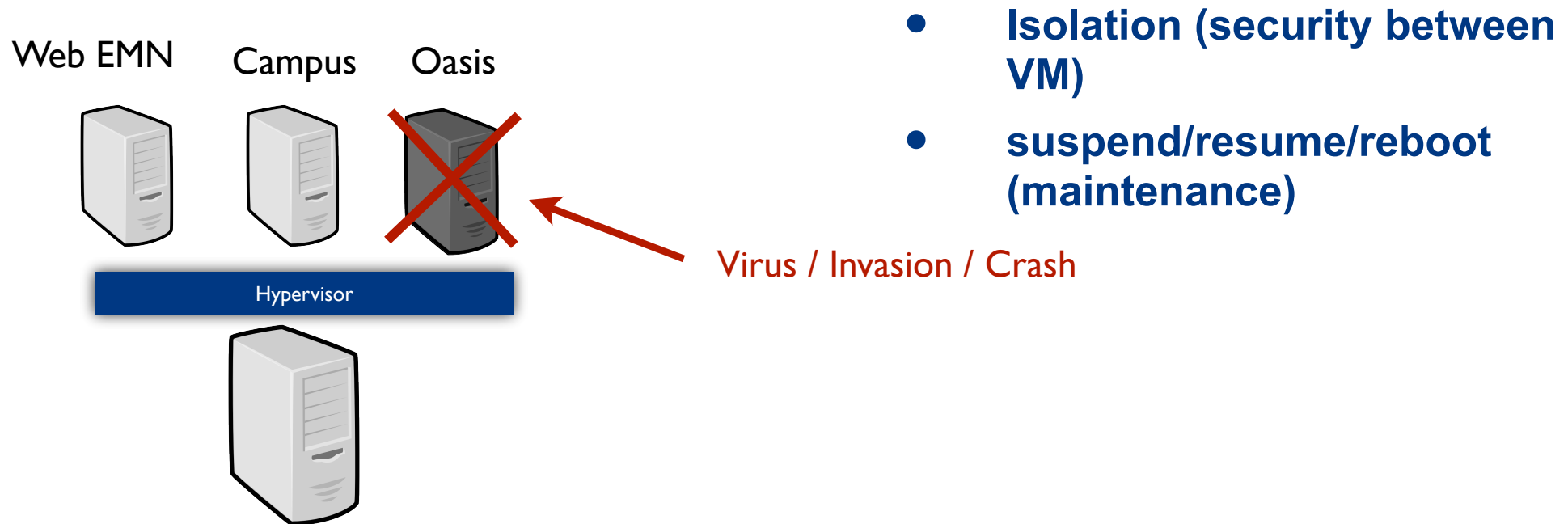
ESXi

## BENEFITS AND FEATURES

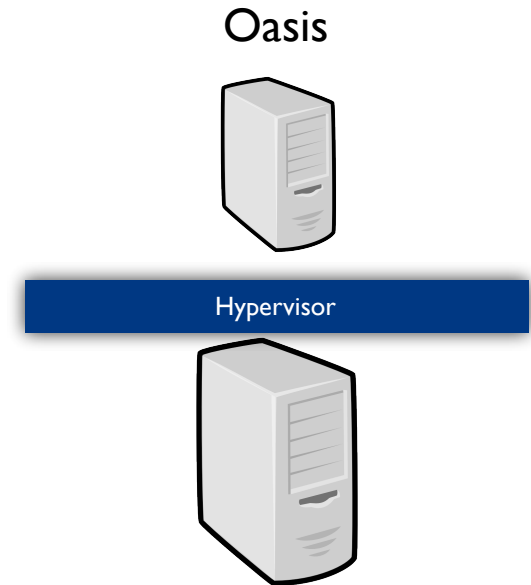
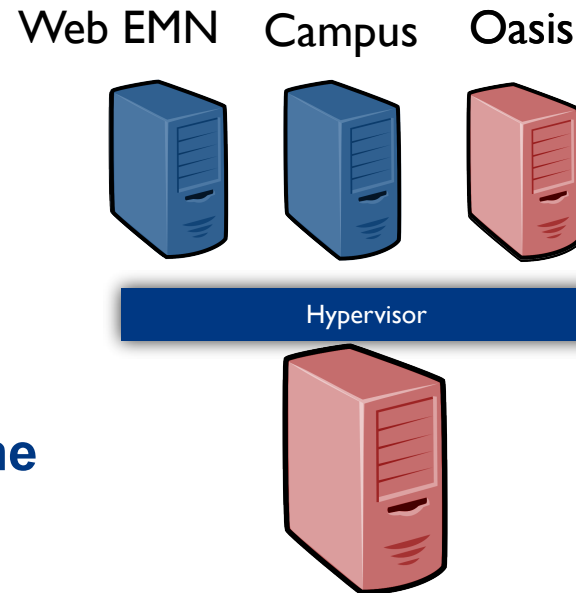
- **Consolidation (virtualization effect) :**
  - Consolidating to virtual machines reduces the number of running nodes  
So energy consumption
  - Reduces hardware costs while providing more efficient node
- **How ? : Virtualisation capabilities**







- **Live migration (load-balancing)**
- **High Availability (downtime ~ 60 ms)**



- **Dynamic Consolidation :**
  - The resources are allocated depending on the VM needs
  - VMs are mixed to be hosted on a reduced number of nodes
  - Servers unused can be turned off
  - VMs are remixed when it is necessary

- Host system protected from VMs, VMs protected from each other
  - I.e. A virus less likely to spread
  - Sharing is provided though via shared file system volume, network communication
- Freeze, **suspend**, running VM
  - Then can move or copy somewhere else and **resume**
  - Snapshot of a given state, able to restore back to that state
    - ▶ Some VMMs allow multiple snapshots per VM
  - **Clone** by creating copy and running both original and copy
- Great for OS research, better system development efficiency
- Run multiple, different OSes on a single machine
  - **Consolidation**, app dev, ...

- **Templating** – create an OS + application VM, provide it to customers, use it to create multiple instances of that combination
- **Live migration** – move a running VM from one host to another!
  - No interruption of user access
- All those features taken together -> **cloud computing**
  - Using APIs, programs tell cloud infrastructure (servers, networking, storage) to create new guests, VMs, virtual desktops

# CHAPTER 4.2

## INSIDE VIRTUALIZATION



**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

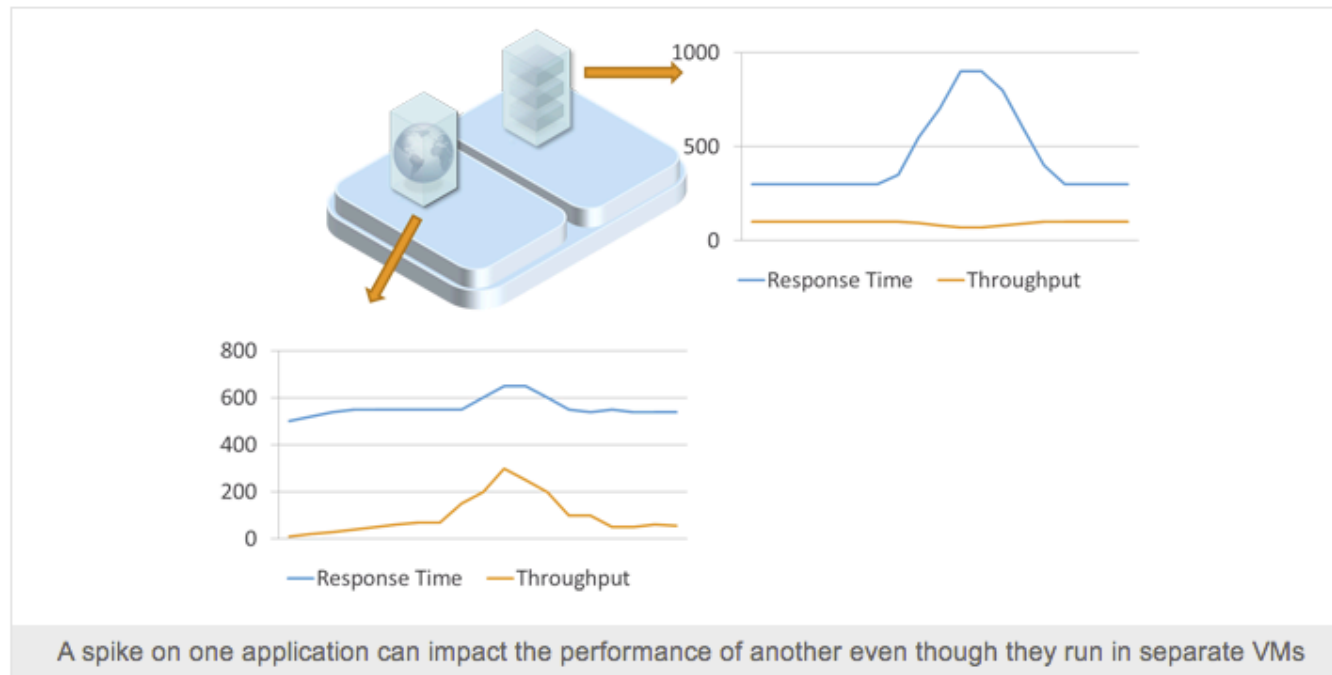
- Now look at operating system aspects of virtualization
  - CPU scheduling, memory management, I/O, storage, and unique VM migration feature
    - ▶ How do VMMs schedule CPU use when guests believe they have dedicated CPUs?
    - ▶ How can memory management work when many guests require large amounts of memory?



- Even single-CPU systems act like multiprocessor ones when virtualized
  - One or more virtual CPUs per guest
- Generally VMM has one or more physical CPUs and number of threads to run on them
  - Guests configured with certain number of VCPUs
    - ▶ Can be adjusted throughout life of VM
  - When enough CPUs for all guests -> VMM can allocate dedicated CPUs, each guest much like native operating system managing its CPUs
  - Usually not enough CPUs -> CPU **overcommitment**
    - ▶ VMM can use standard scheduling algorithms to put threads on CPUs
    - ▶ Some add fairness aspect

- Cycle stealing by VMM and oversubscription of CPUs means guests don't get CPU cycles they expect
  - Consider timesharing scheduler in a guest trying to schedule 100ms time slices -> each may take 100ms, 1 second, or longer
    - ▶ Poor response times for users of guest
    - ▶ Time-of-day clocks incorrect
  - Some VMMs provide application to run in each guest to fix time-of-day and provide other integration features

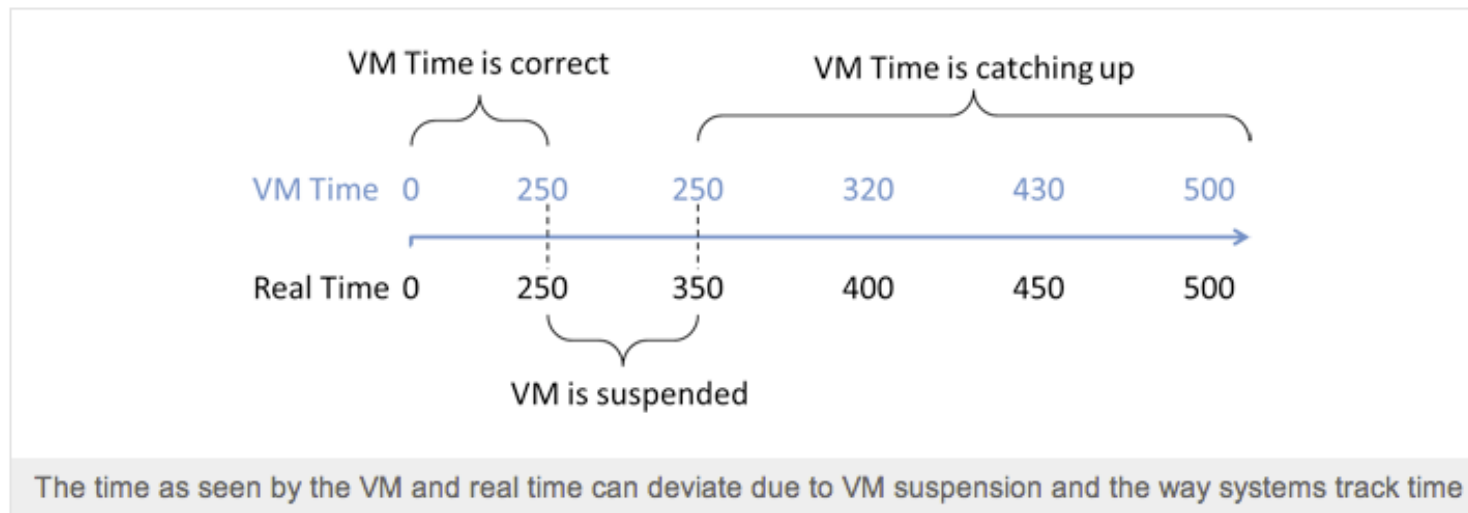
## Why running isolated VMs also hides the root cause of performance problems



A problem caused by an application running in one VM, only shows up in the second VM

We need to correlate the performance measurements from both applications using monitoring data at the hypervisor layer

When the hypervisor suspends a guest system, it is just like an OS suspending one process to execute another. But where the OS is responsible for time keeping, the guest system won't usually know when it is suspended and can't account for this when tracking time



Windows and Linux systems track time by counting the periodically triggered interrupts.

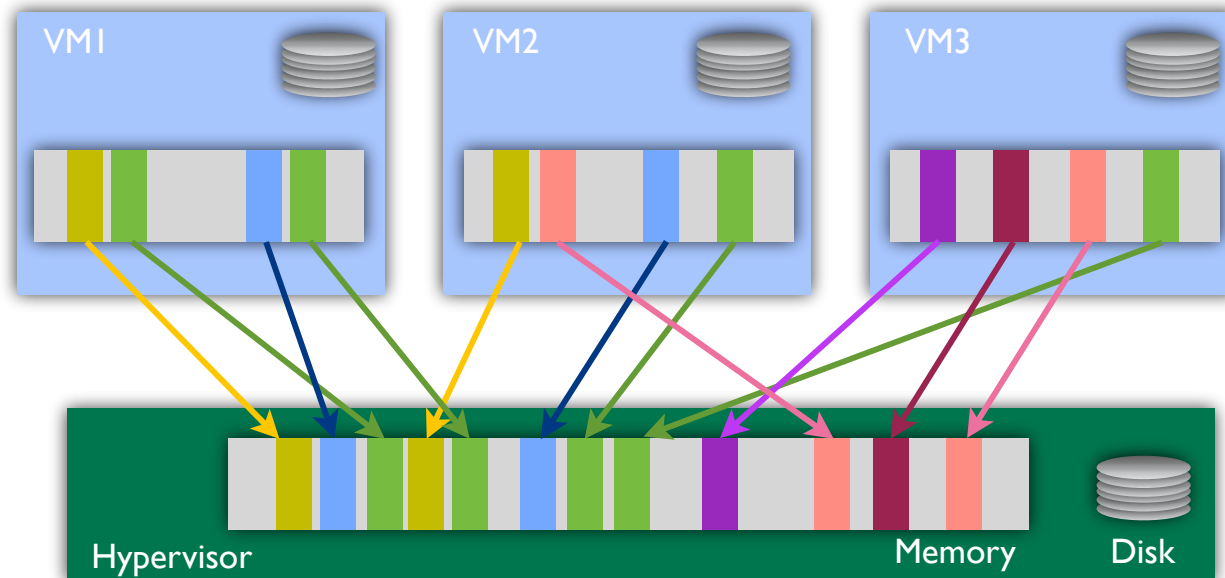
A suspended guest can't count interrupts

The hypervisor records interrupts and replays them when the VM is resumed.

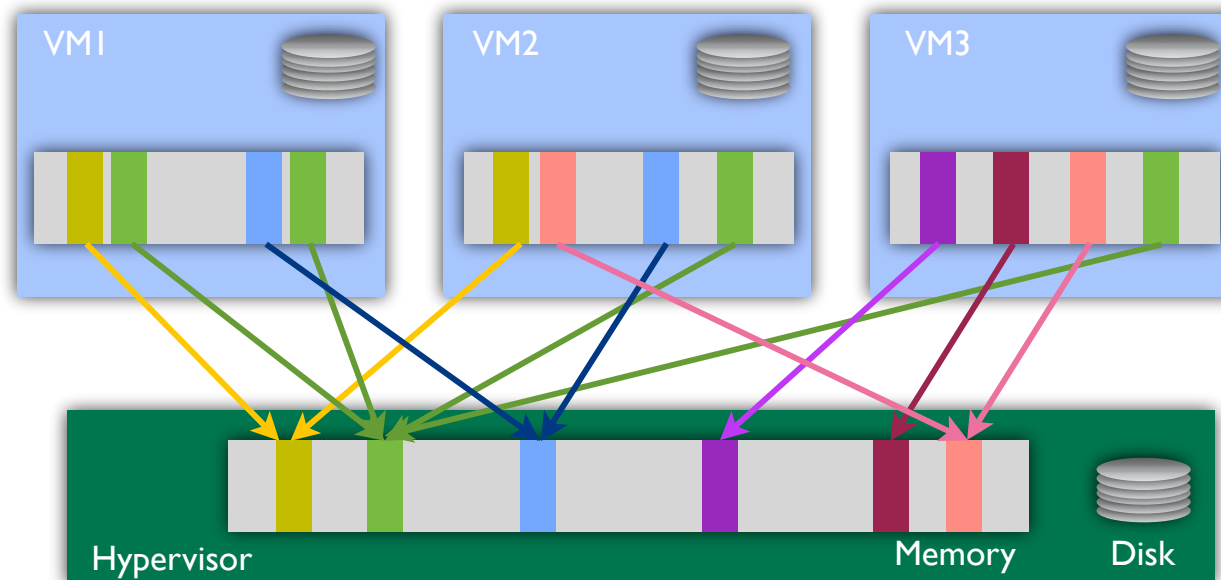
After resuming, a lot of queued interrupts are processed in quick succession.

As the interrupts represent time slices, time effectively speeds up!

## Memory overcommitment



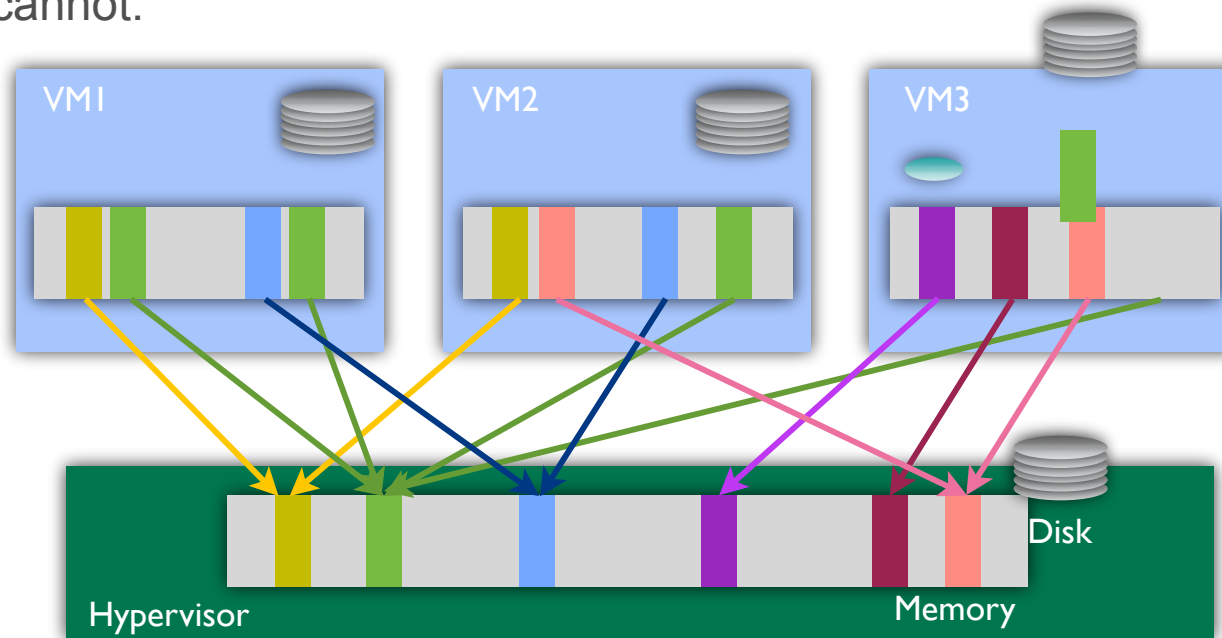
## Content-Based sharing



- The concept of transparent page sharing was first proposed in the Disco system [1997-17]

## Ballooning

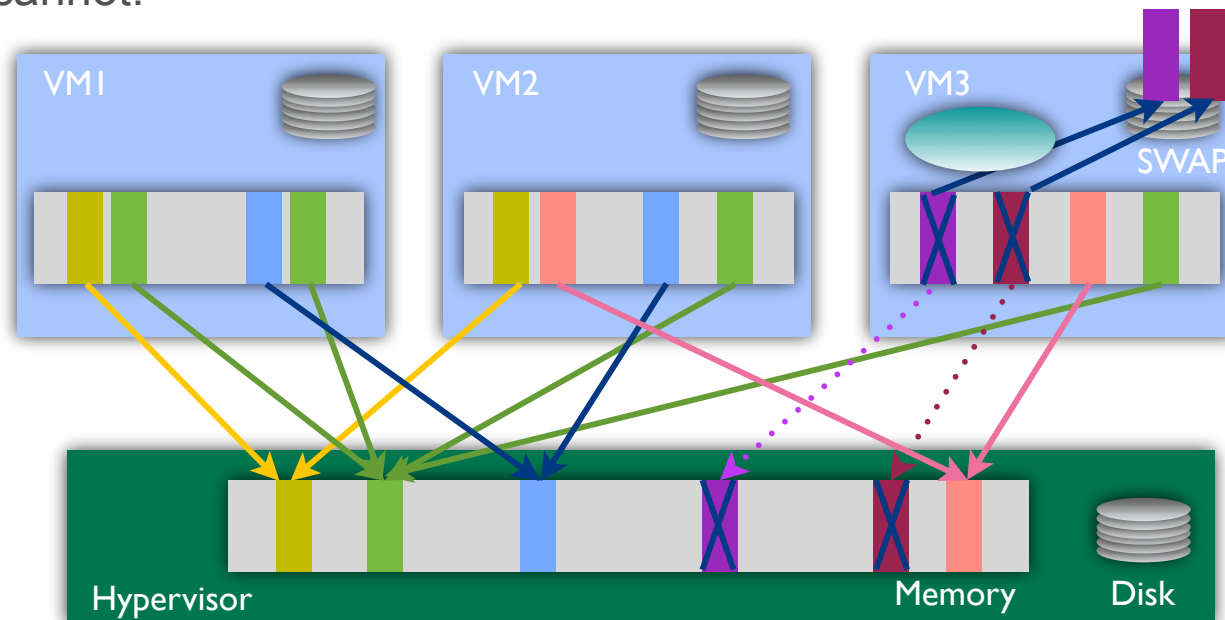
The guest can make an informed decision about what to swap, whereas the hypervisor cannot.





## Ballooning

The guest can make an informed decision about what to swap, whereas the hypervisor cannot.

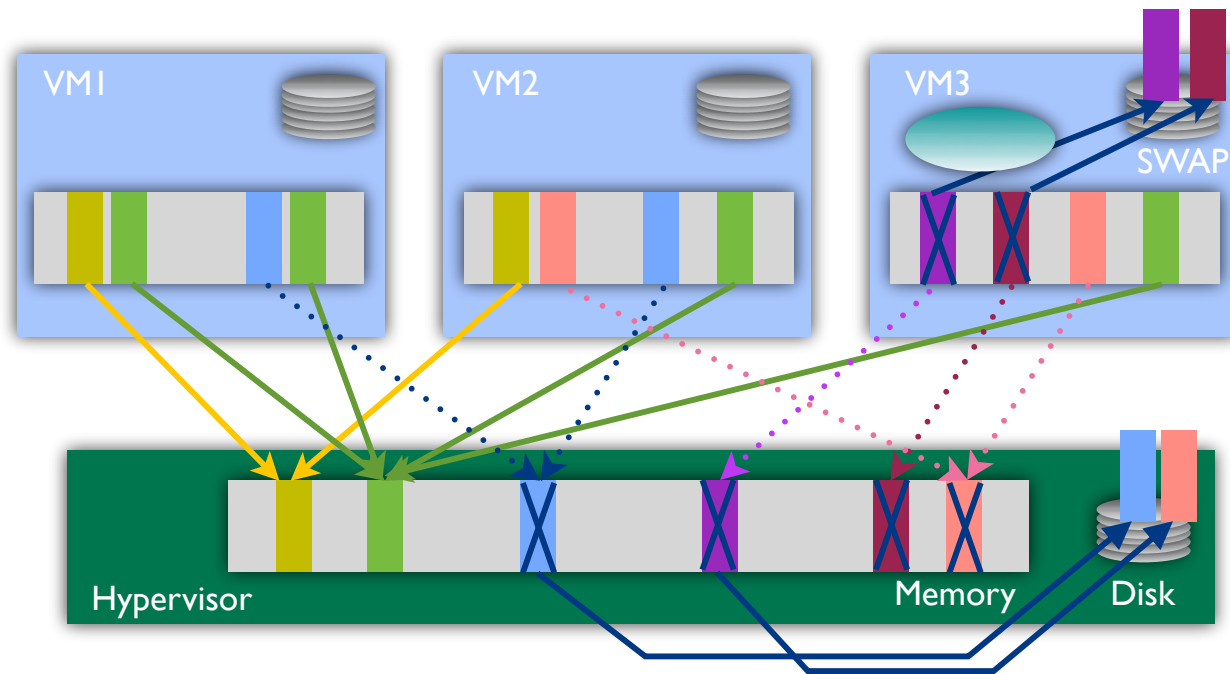


Inflate the balloon by allocating guest memory and pinning (can not swapped to disk) the underlying memory pages.

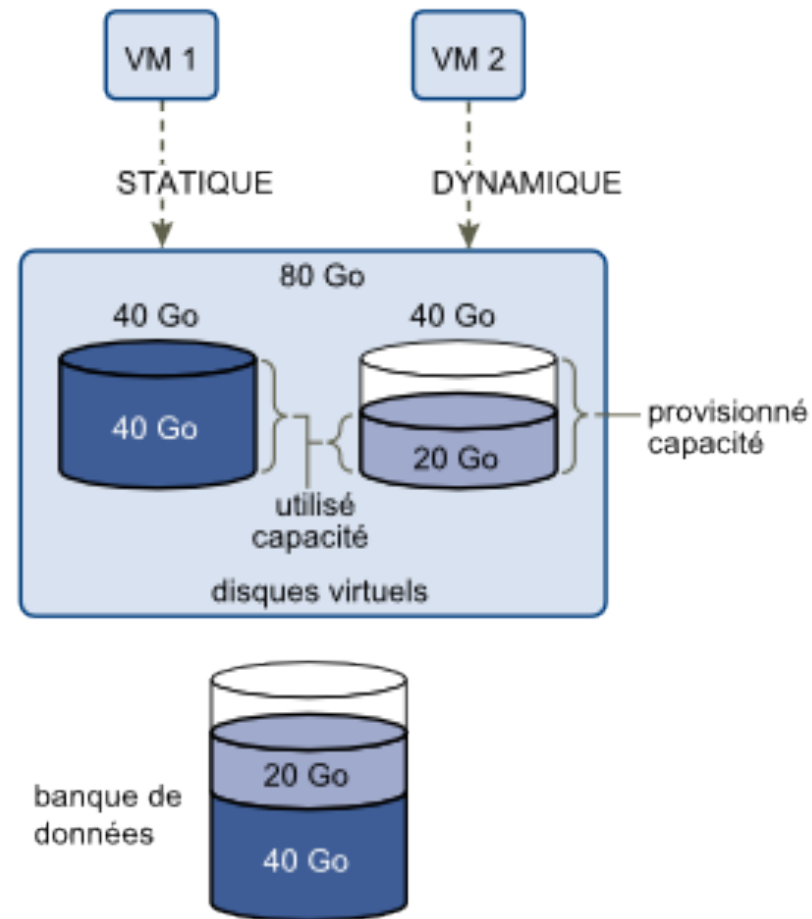
The hypervisor can reclaim those pinned pages and reassign it to another VM.

A guest with a balloon might be forced to swap other pages.

# Hypervisor Swapping

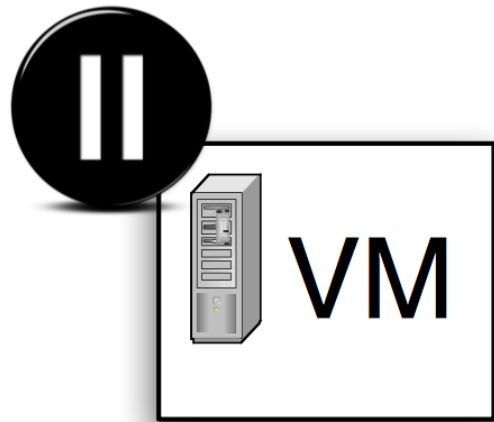


- **Effective only if it is complemented by algorithms that ensure that the VMs resident on each physical server contain a significant amount of sharable pages.**
- **Memory Buddies [2010] Goals :**
  - Analyze the memory contents of multiple VMs to determine **sharing potential** then find **more compact VM placement**
  - Evaluation show that “sharing aware” placement has the *potential* to significantly improve memory usage (20 VM on 4 servers).
  - **Invasive** system (nucleus component into each virtual machine)
- **Sharing-Aware Algorithms for Virtual Machine Colocation [2012]**
  - *simulation* with (124 VM on 25 servers) and **offline**
- **CBS Challenge :**
  - Transparent Page Sharing with Large Pages, Effects of Memory Randomization, Sanitization and Page Cache on Memory Deduplication ...
- **Dynamic consolidation with resource sharing aware**



- Relocate VM from one physical host to another  
Complete encapsulation → no OS support needed
- Transfer VM state over the network
  - Processor state (CPU registers)
  - Hardware devices state (hardware registers)
  - Memory content  
(Possibly disk content)

- Simplest approach
- Suspend source VM on source host
- Copy all VM state over the network
- Resume source VM on destination host
- Used by the Internet Suspend/Resume project

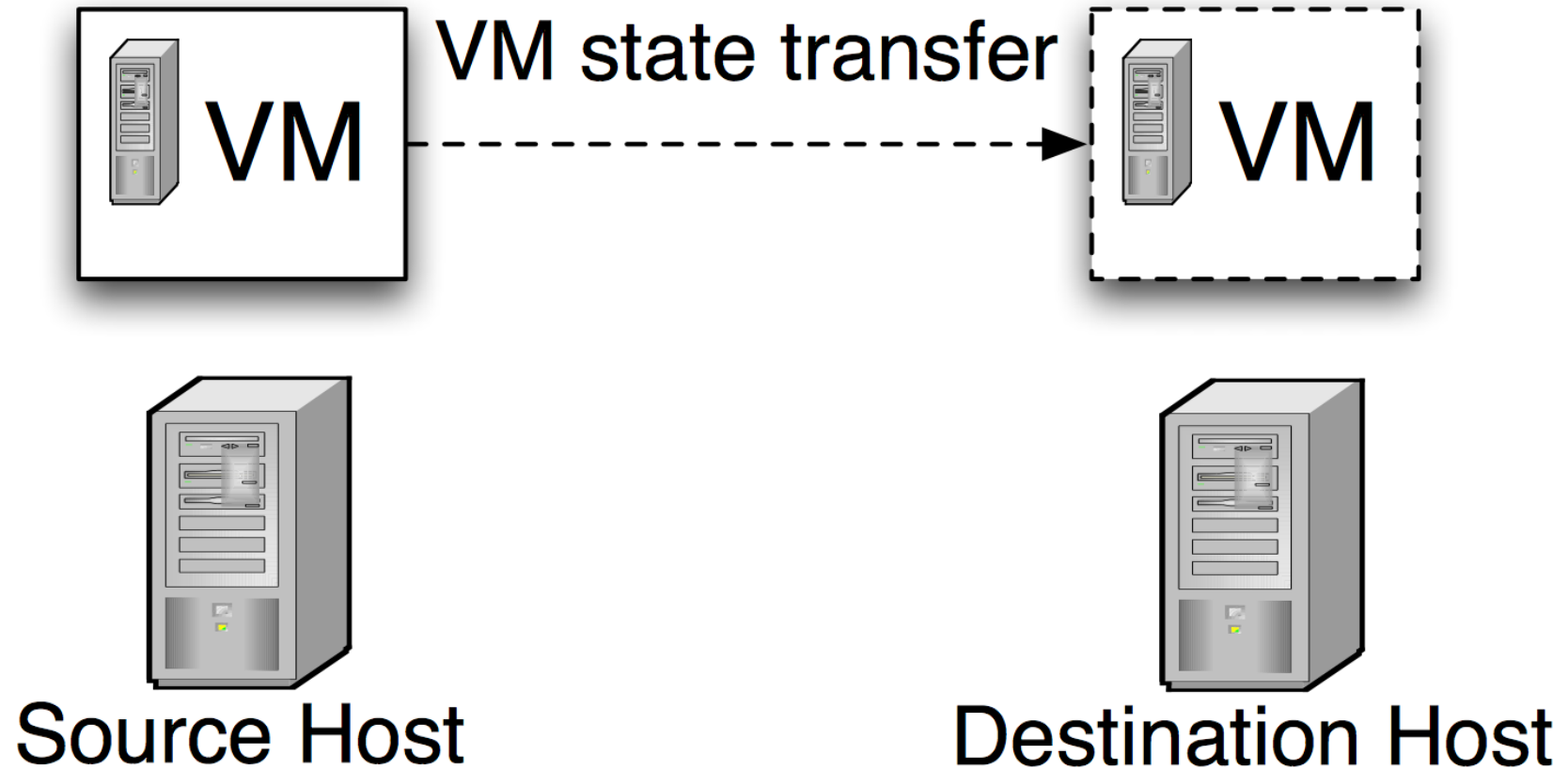


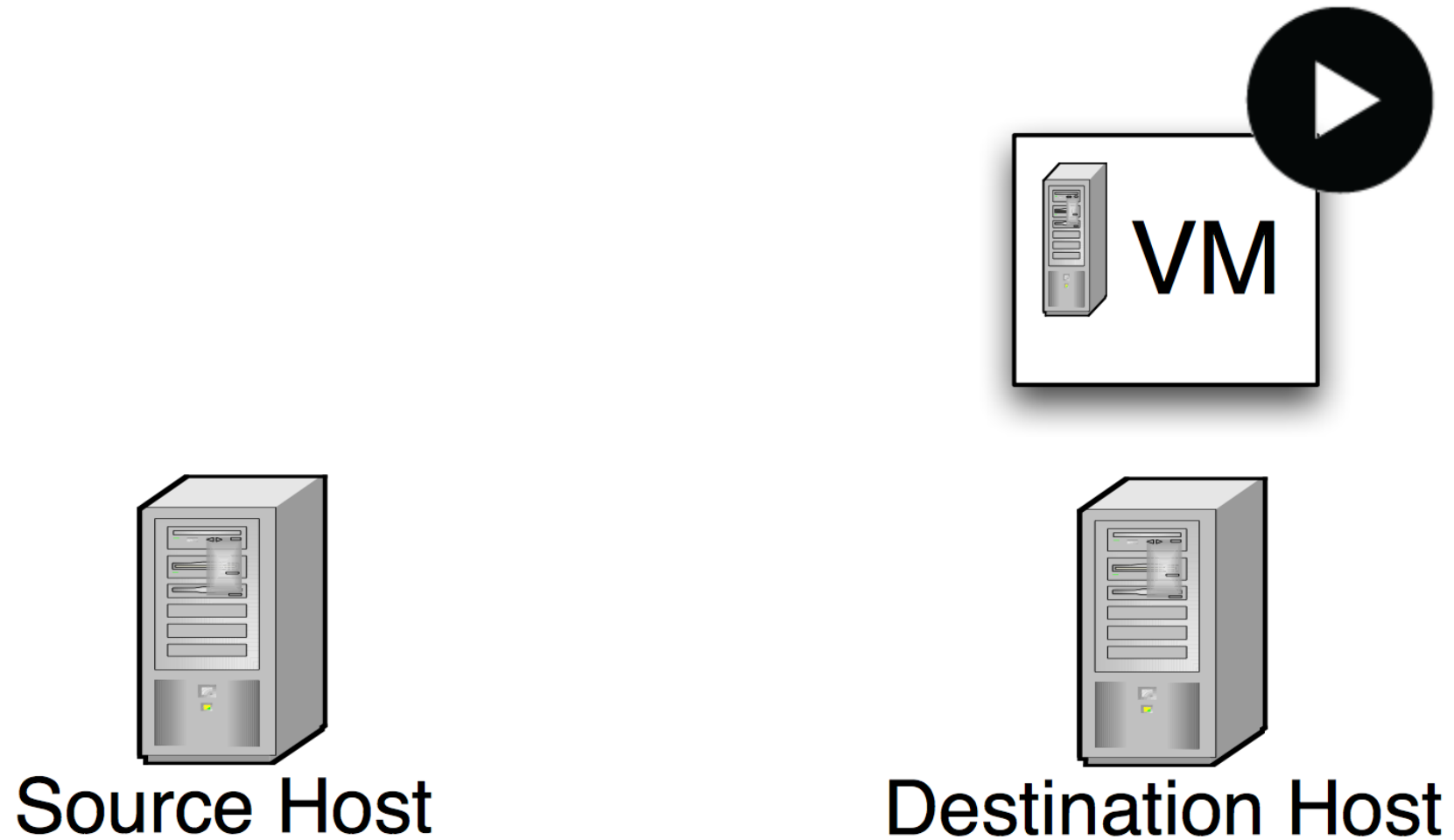
Source Host



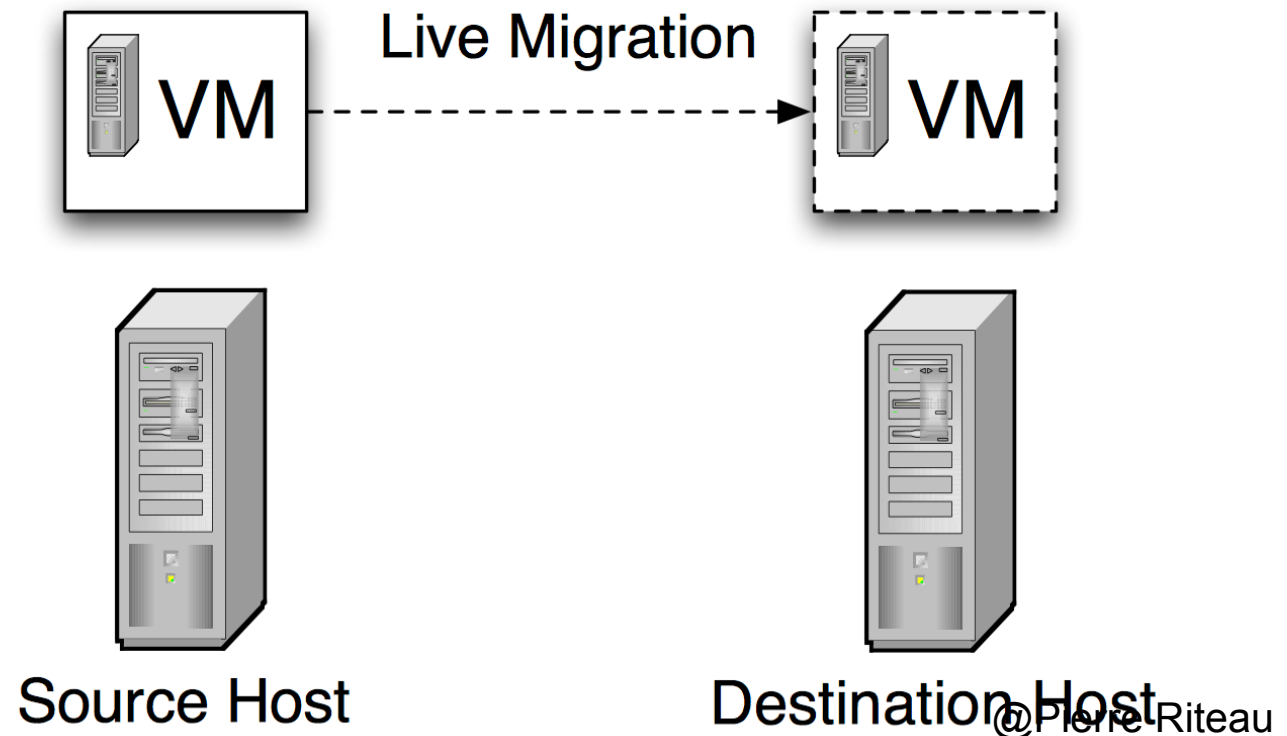
Destination Host



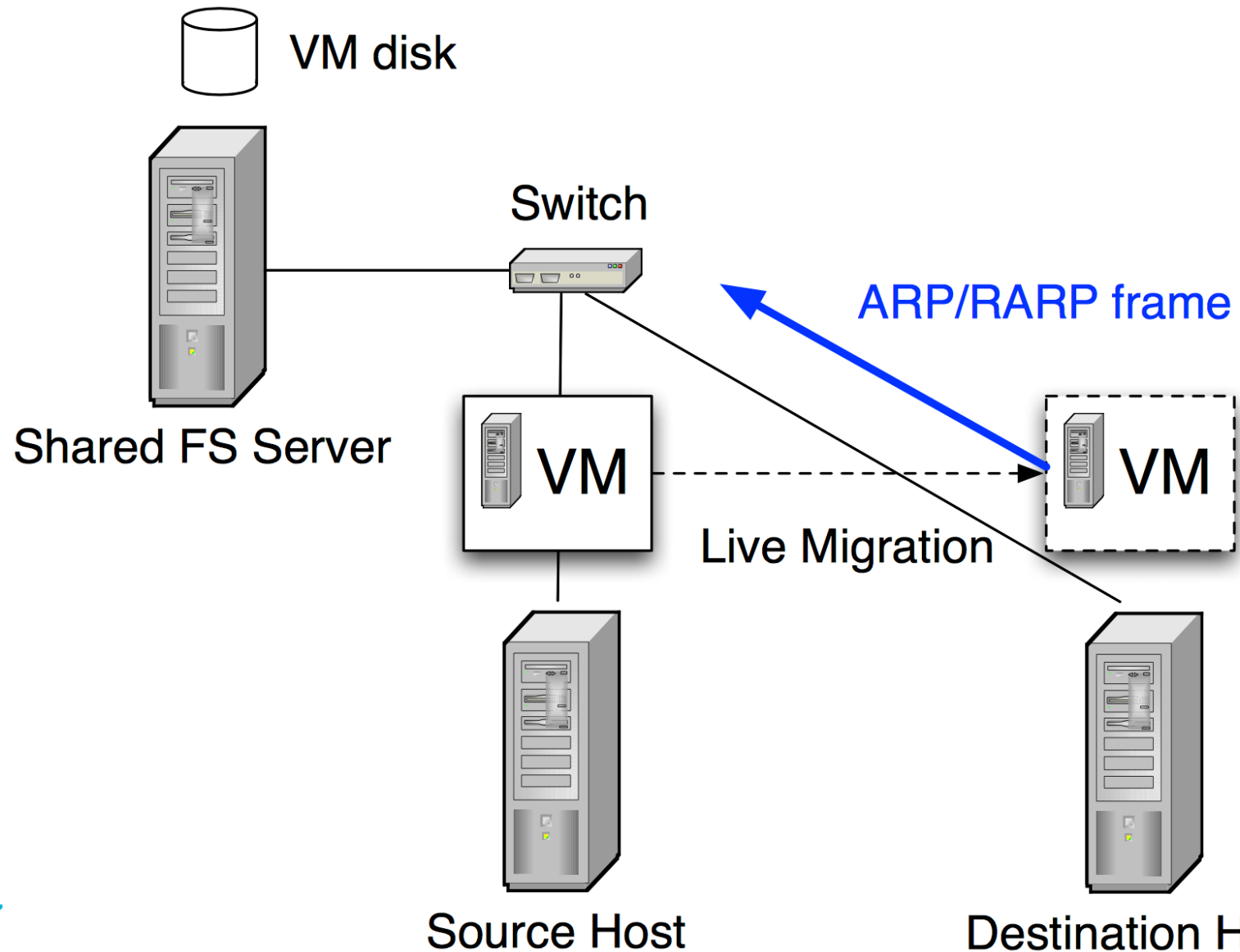




- Problem with pure stop-and-copy:
  - long downtime
- Live migration
  - Minimize downtime (milliseconds)
  - Works by transferring state during execution

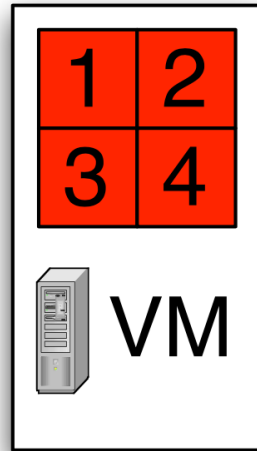


- Live migration initially proposed for LANs
- Clark et al., NSDI '05 & Nelson et al., USENIX '05
- Transfer from source host to destination host of the same LAN
- What about storage and network resources?
  - Shared storage (e.g. NFS) → no migration needed
  - Network traffic redirected with gratuitous ARP/RARP frames



- Offers many advantages
  - Load balancing / Reduced energy consumption
  - Migrate VMs in case of hotspots
  - Consolidate VMs on a subset of nodes
  - Turn off unused physical nodes
  - Entropy (Jean-Marc Menaud)
- Transparent infrastructure maintenance
- Pro-active fault tolerance
  - Detect future faults from hardware events
  - Preemptively migrate on another node
  - Nagarajan et al., SC 07

- Traditional method used for migration of processes
- Iterative process
  - Copy all memory content to the destination host (while the VM continues running)
  - Do multiples iterations to copy modified memory pages during the previous period
  - When enough iterations have been done, stop the VM and
    - Copy the remaining modified memory pages
    - Copy the CPU and device state
- Resume VM on destination host
- **Method implemented by all production hypervisors**

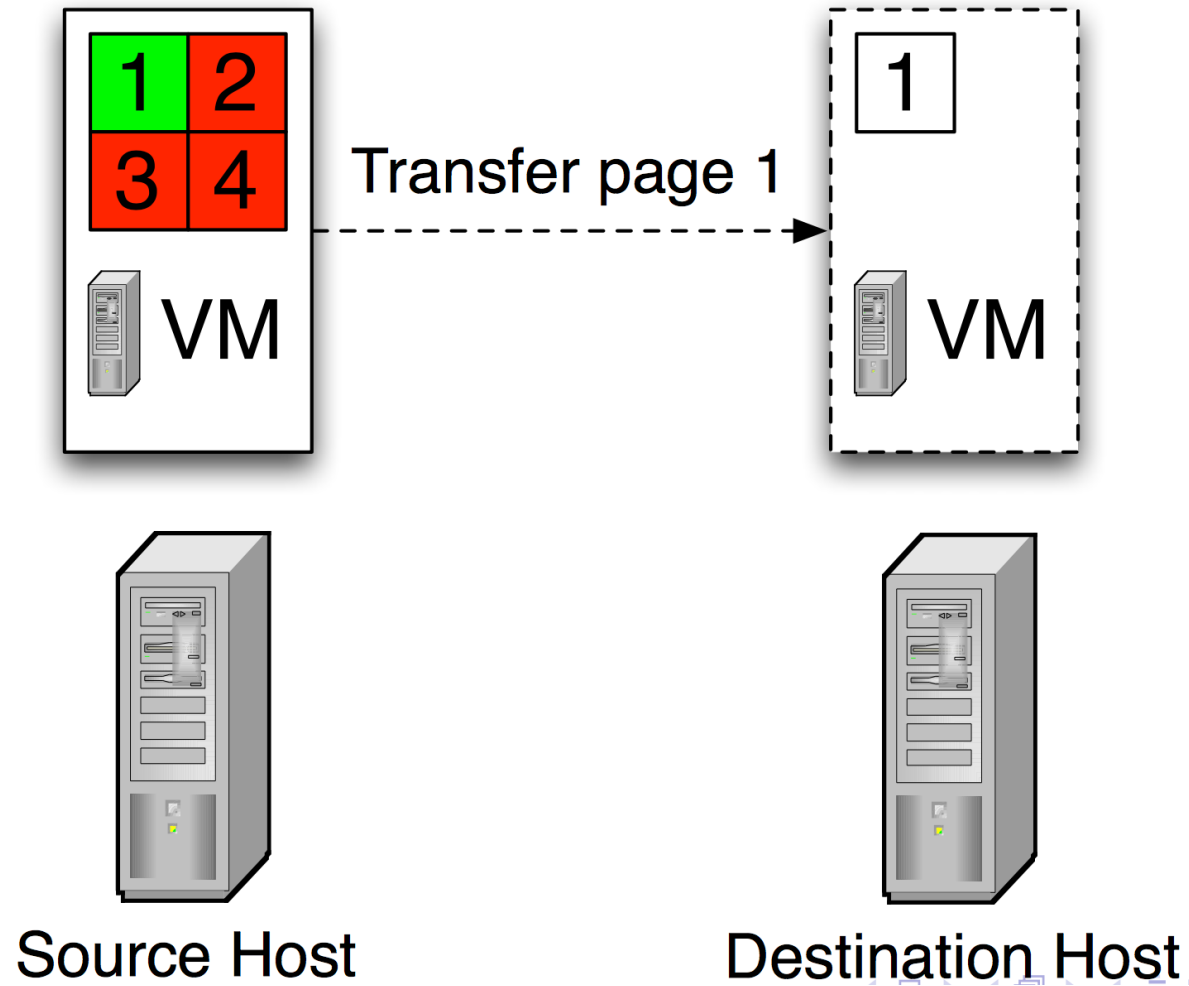


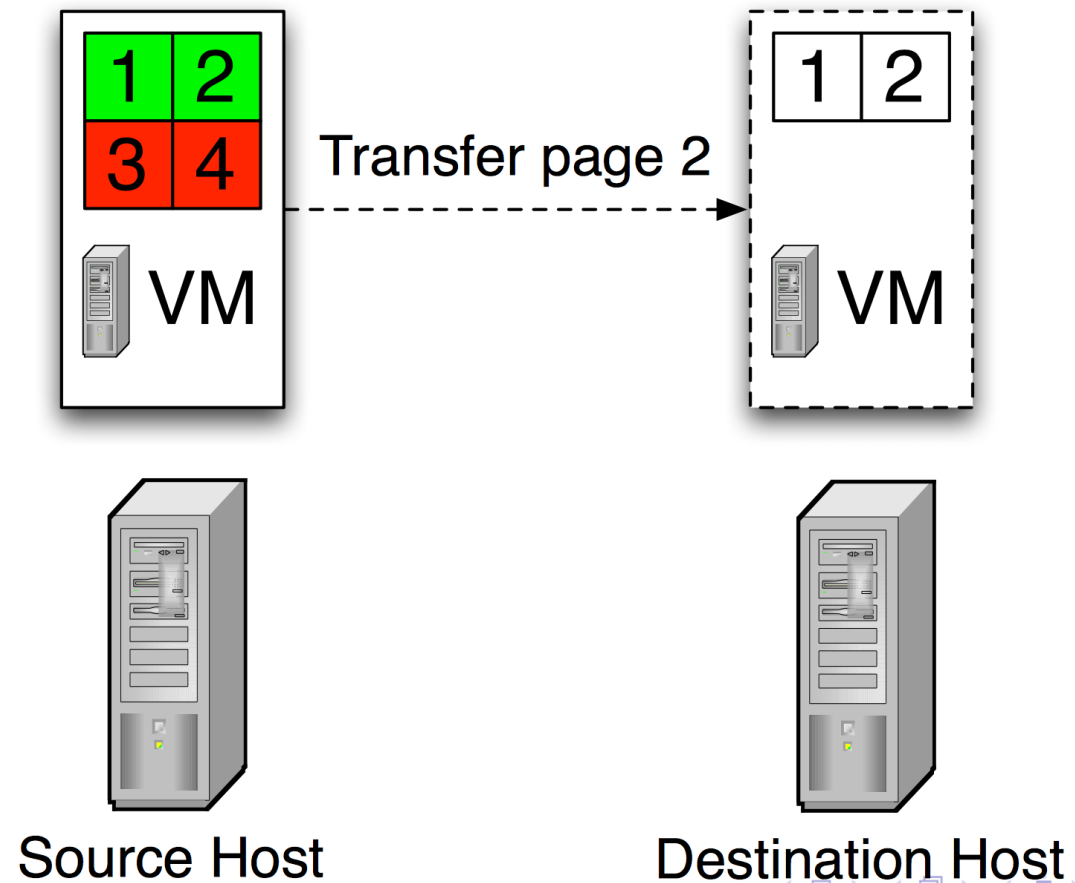
Source Host

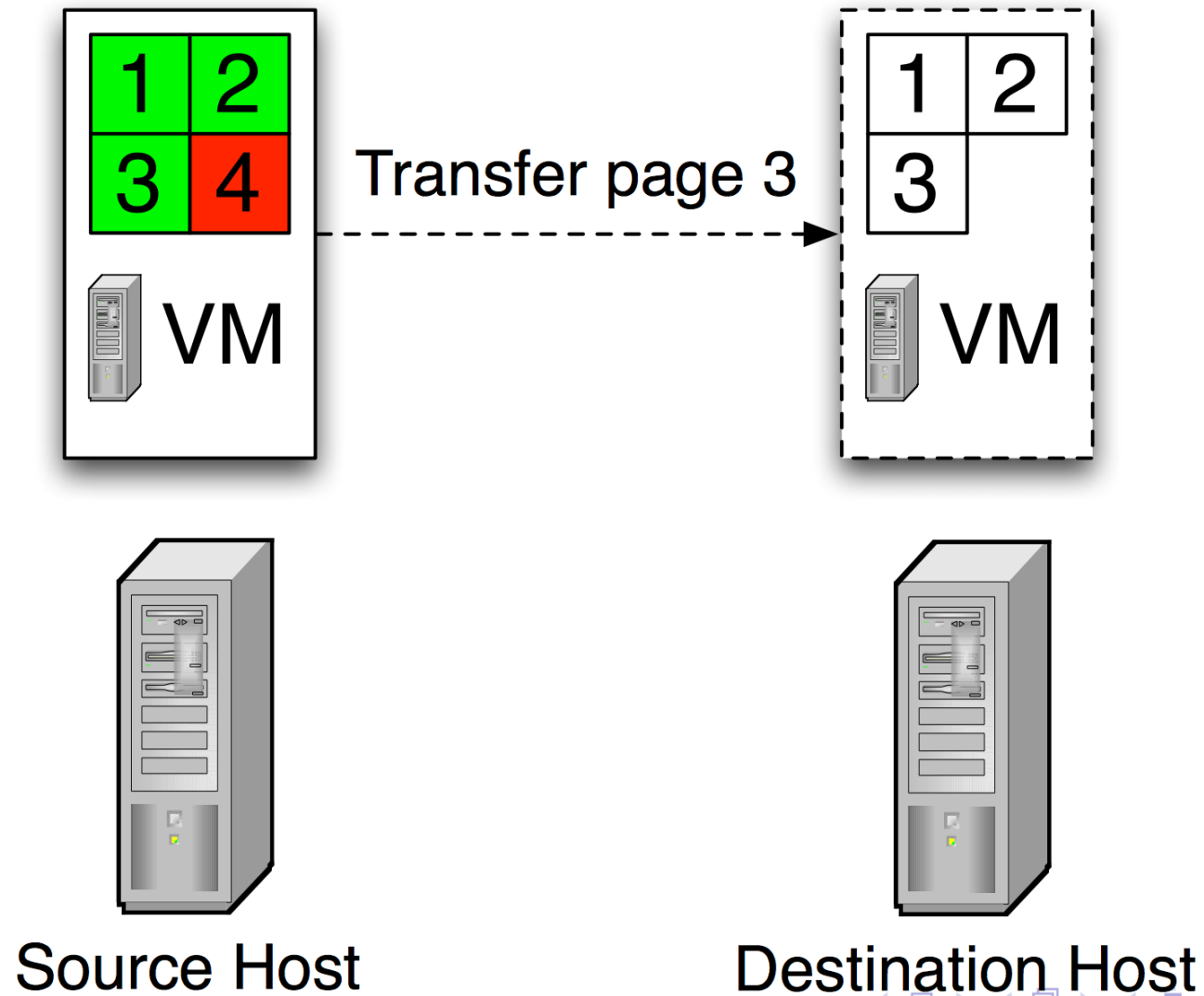


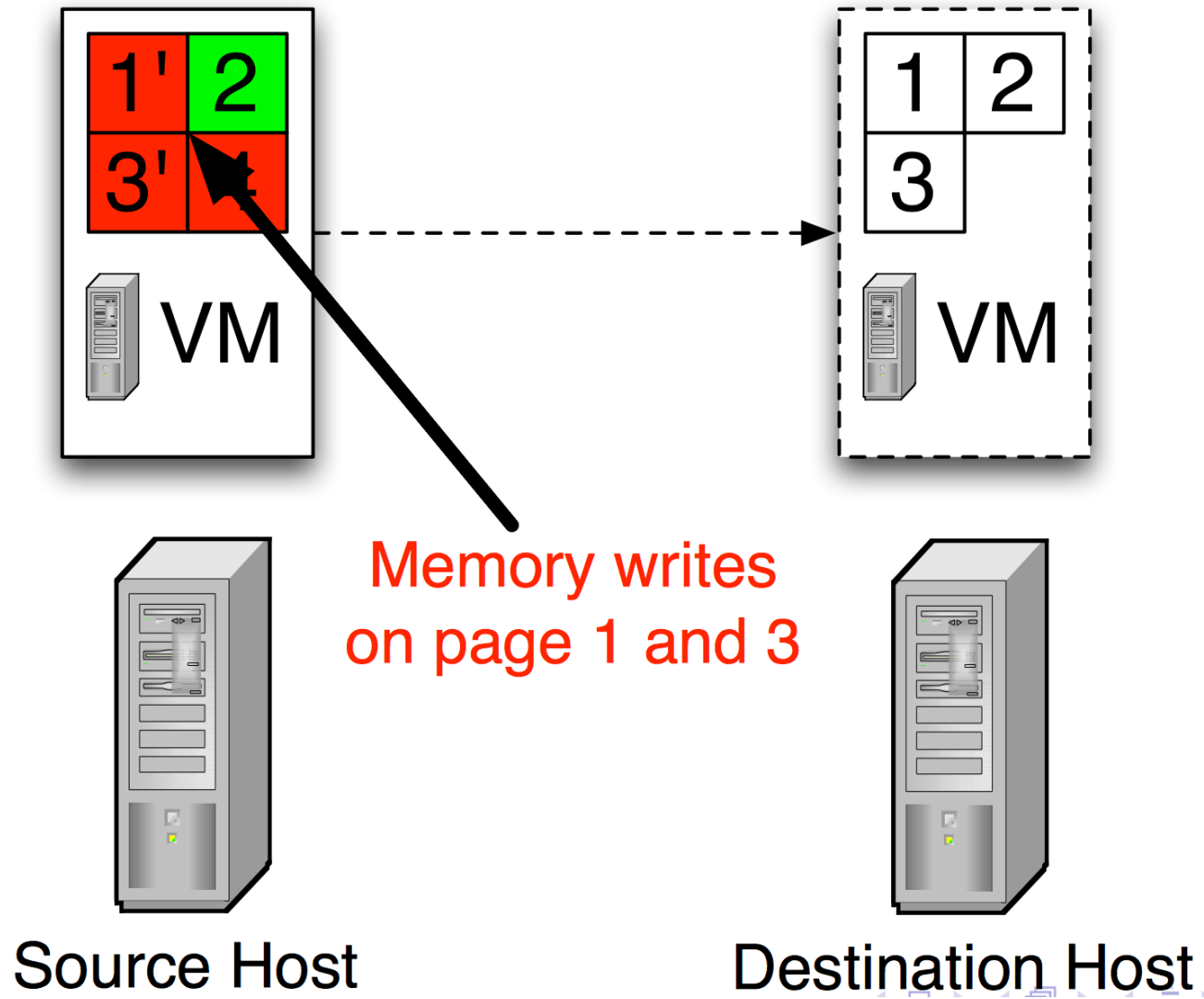
Destination Host

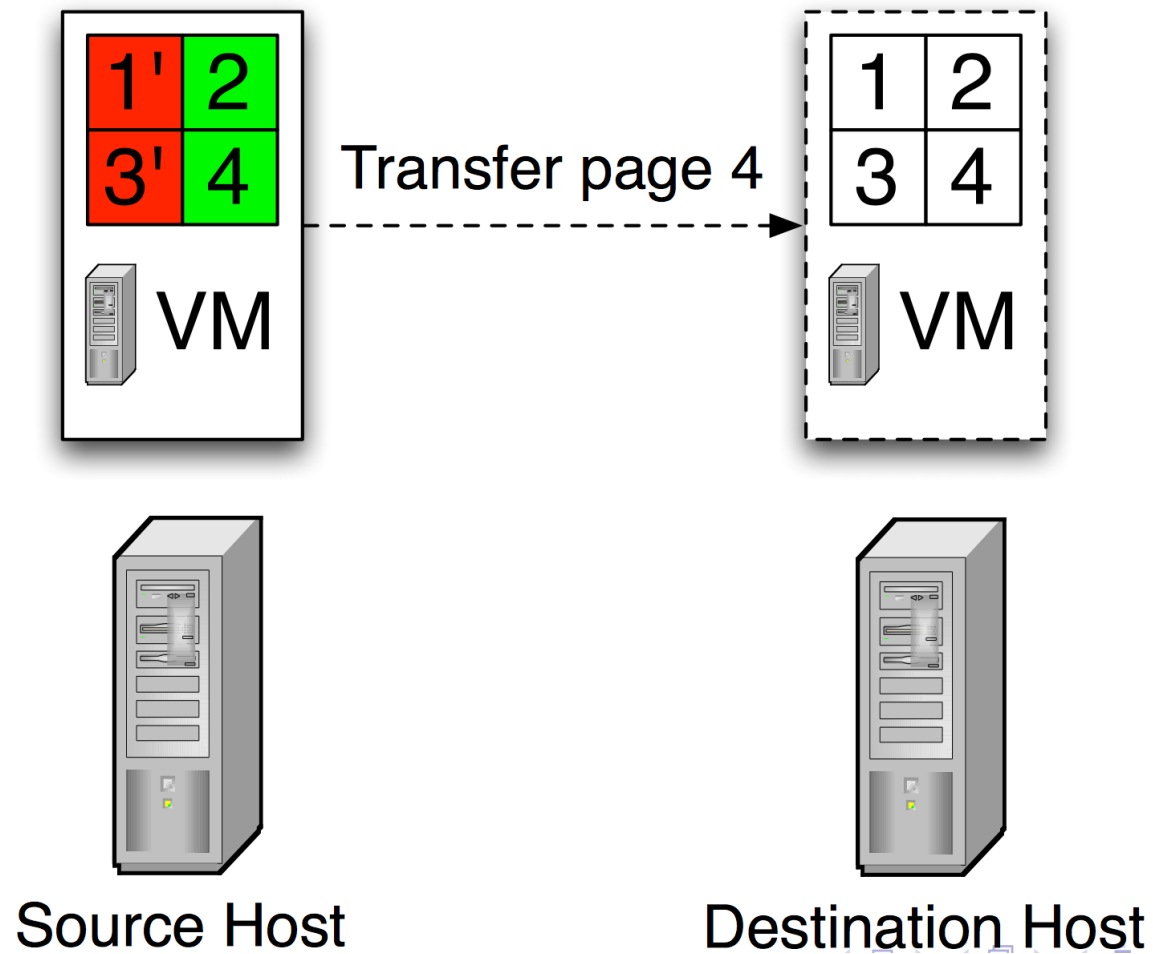


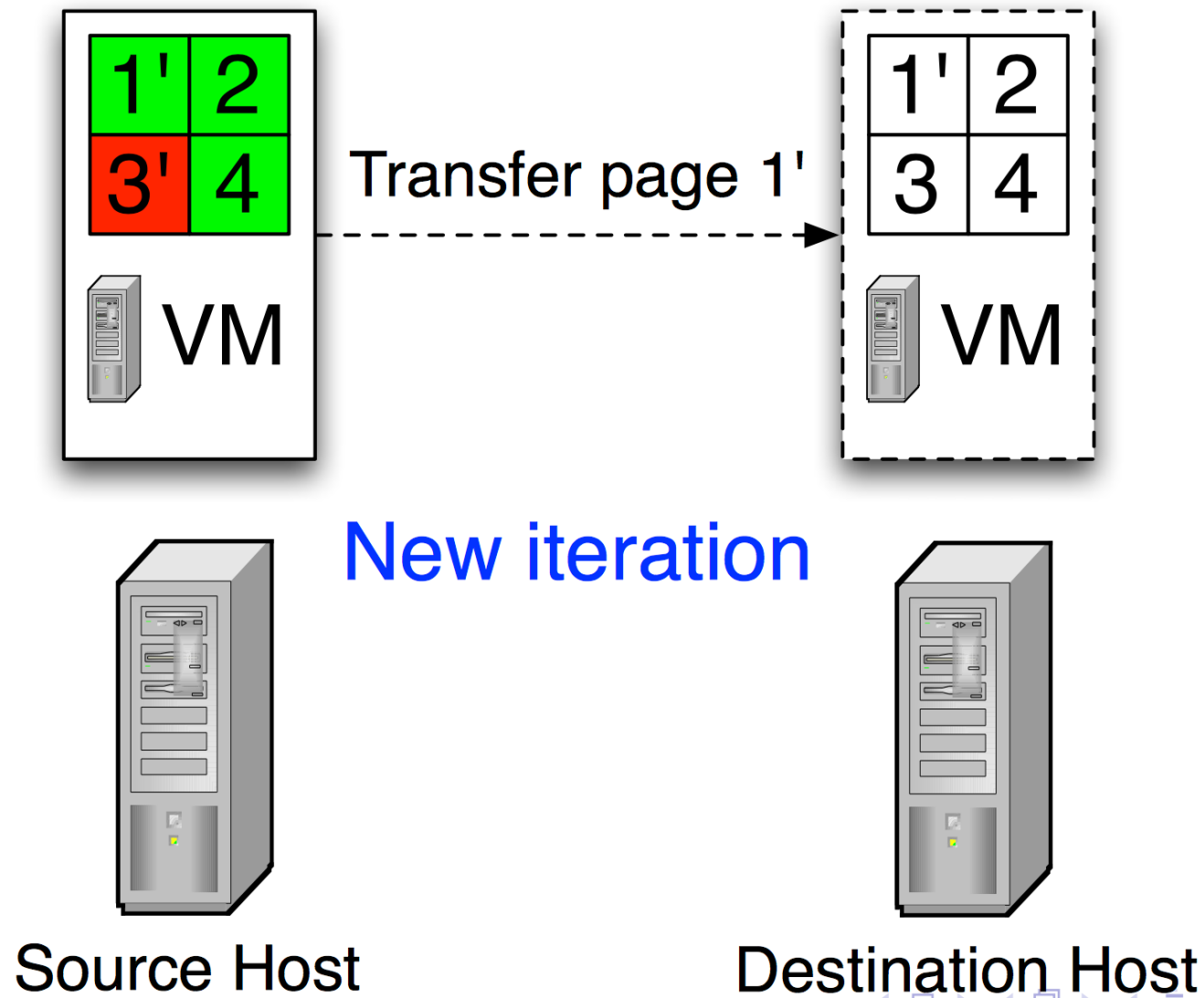


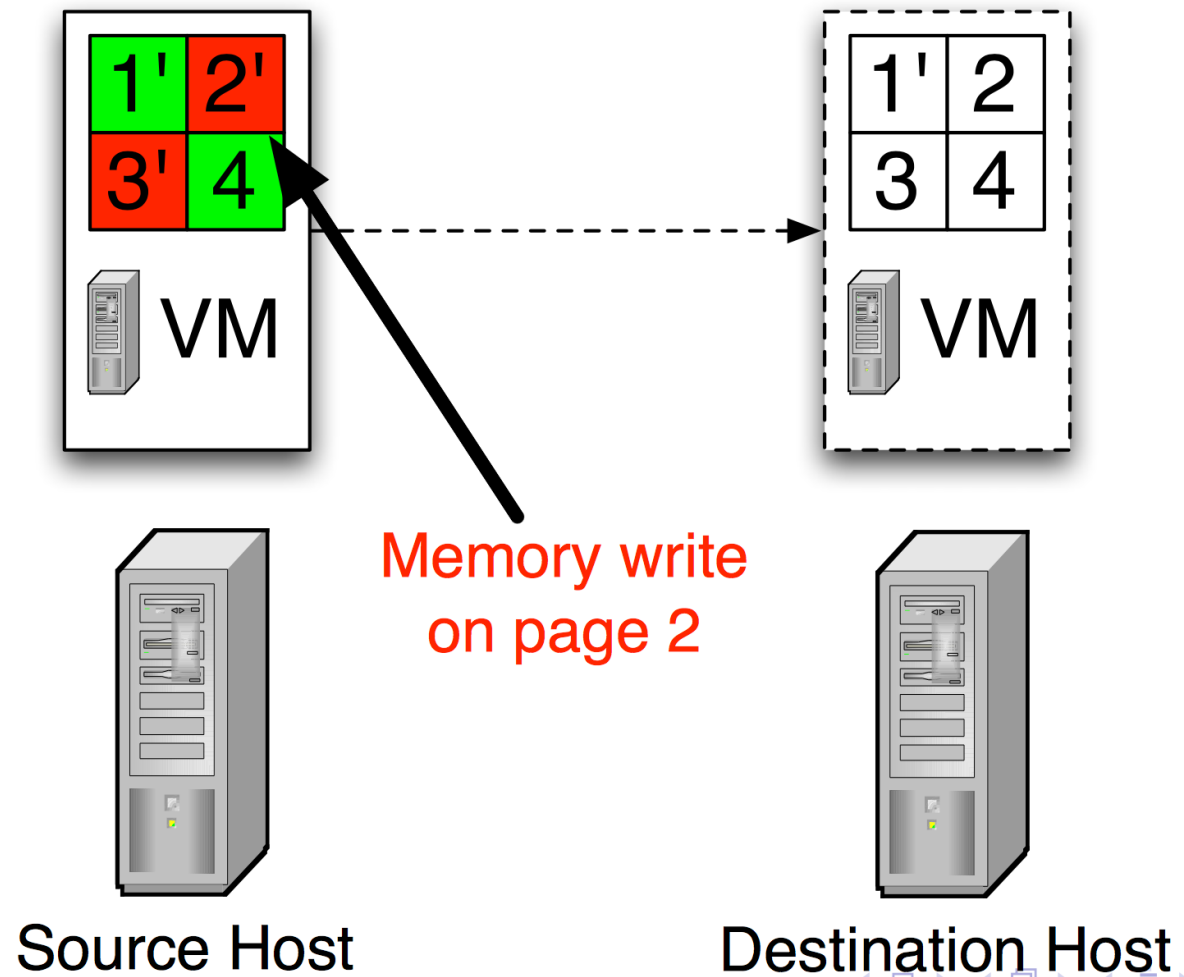


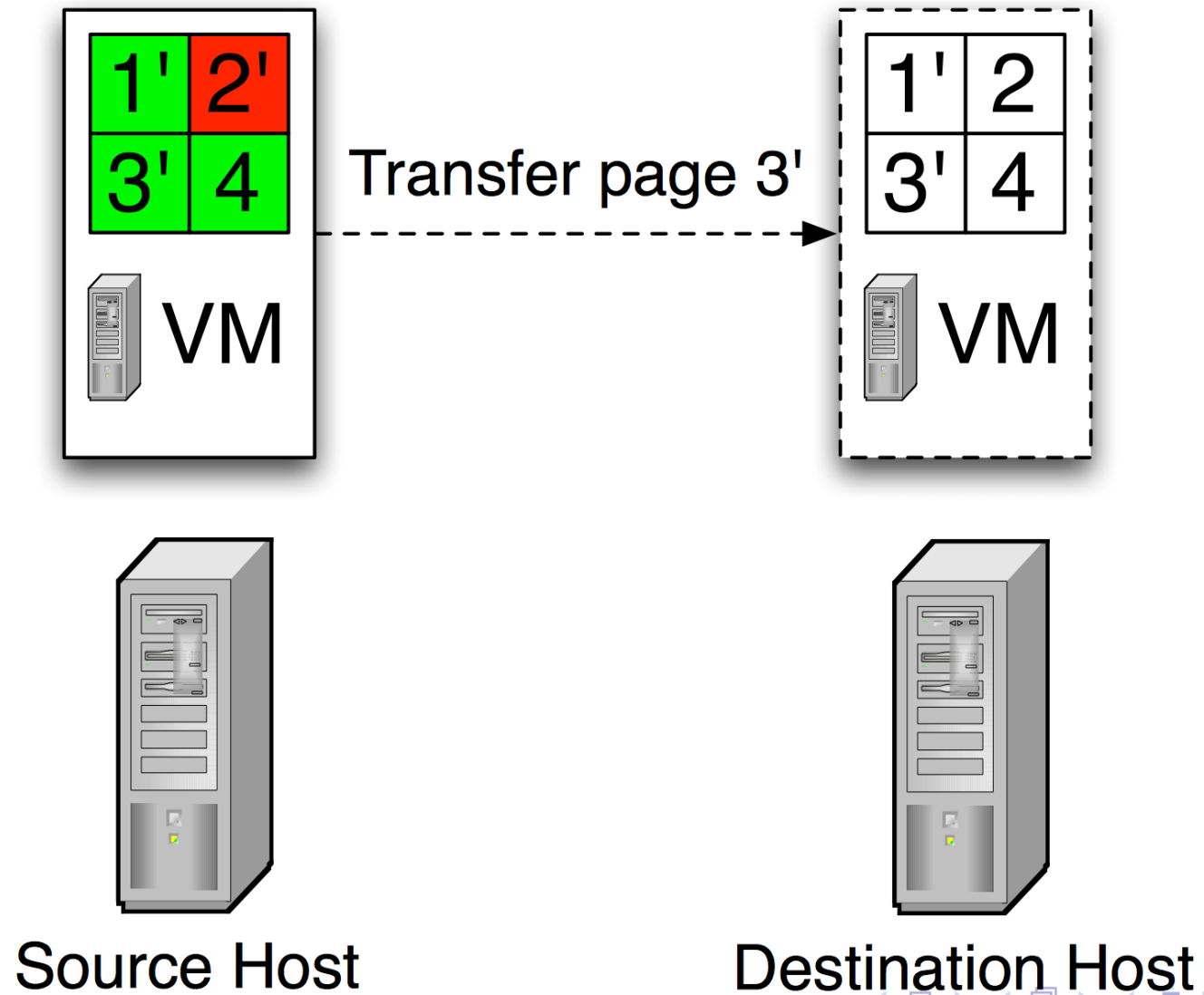




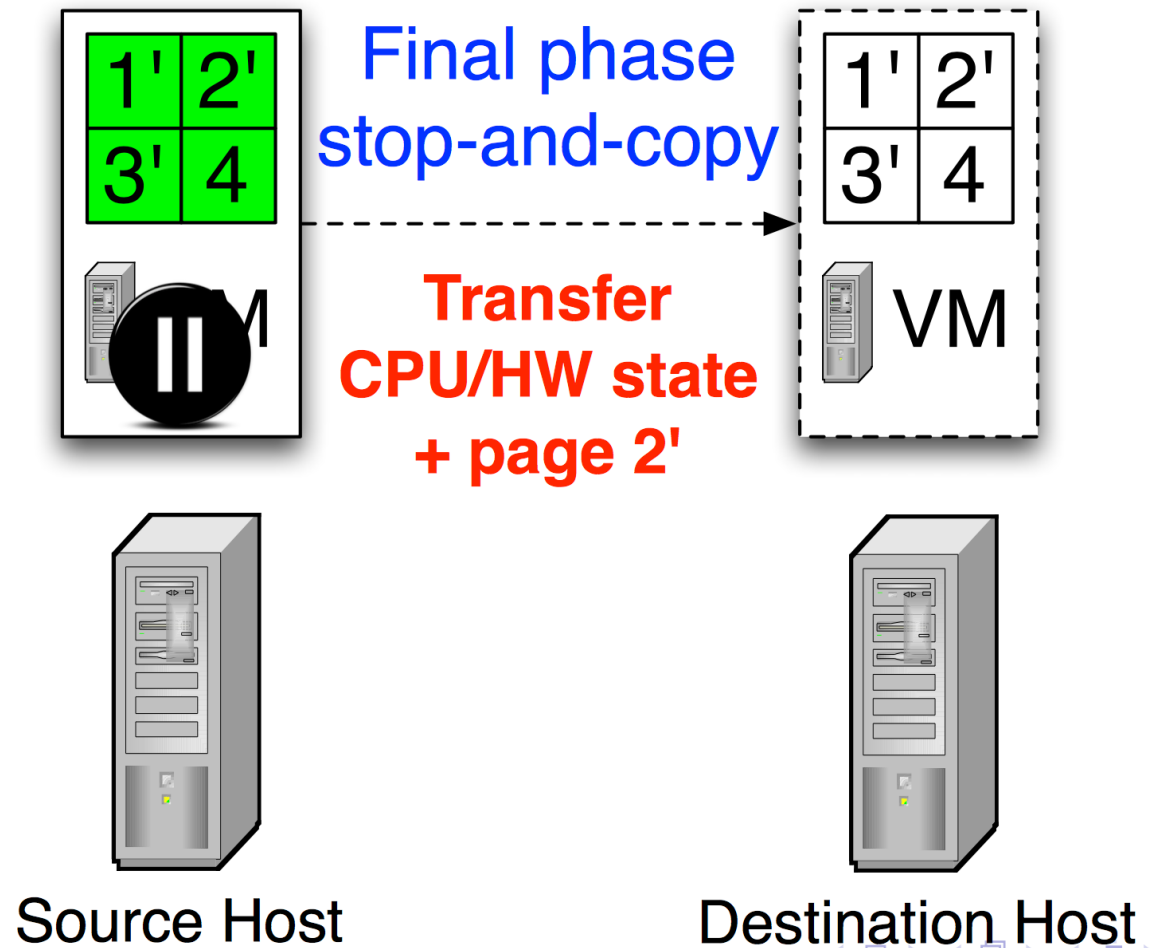


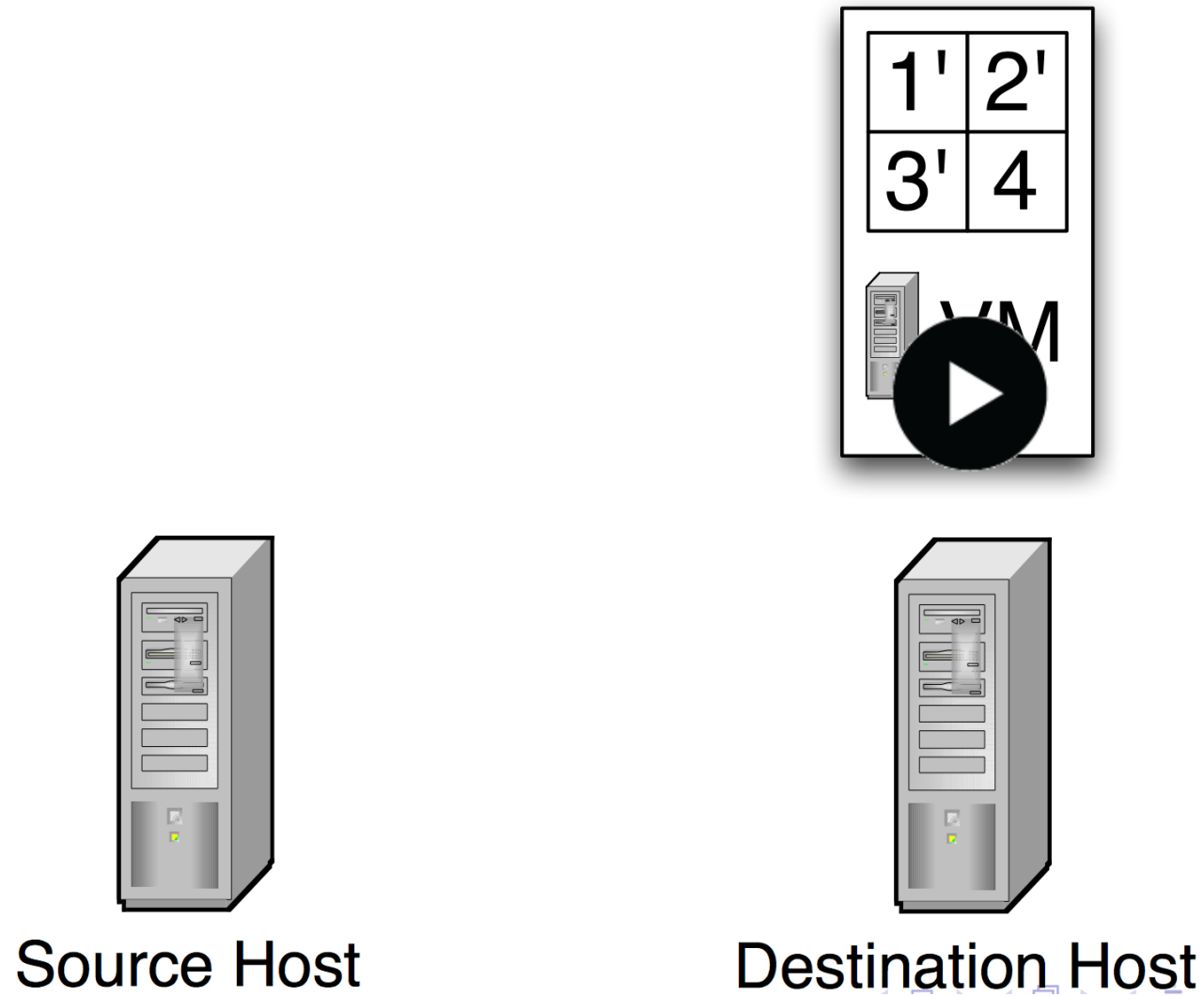










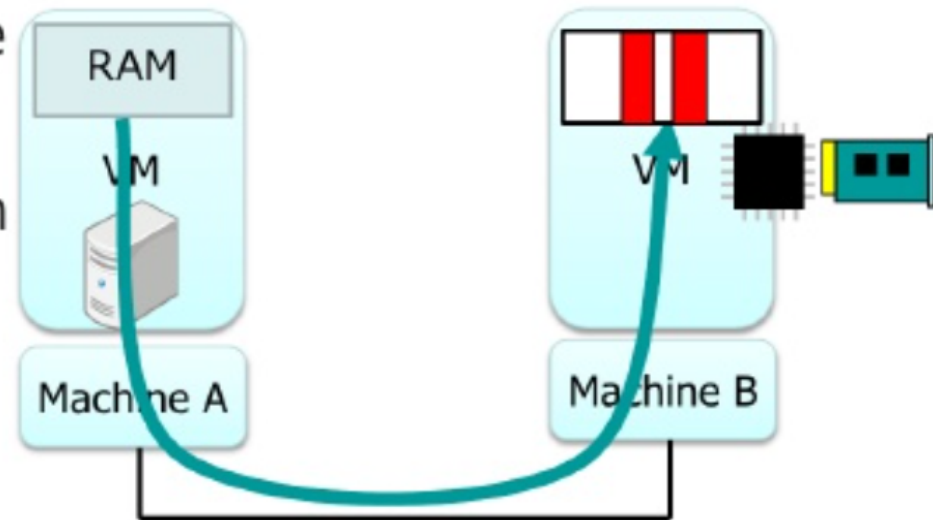


- Different behavior between Xen and KVM
- Xen: threshold values
  - Remaining pages under a threshold
  - OR Too many iterations
  - OR Too much data transferred
- KVM: estimated downtime
  - Administrator can specify maximum downtime
  - Default: 30 milliseconds
  - KVM estimates available bandwidth
  - Stops only when estimated downtime  $<$  maximum downtime
- Xen forces convergences of migration
- KVM trusts the administrator or VM management software

- Pre-copy can present long downtime in the last phase
  - if the application modifies a large working set
  - if the available bandwidth is low
- Post-copy algorithm
  - Start by copying CPU and device state
  - Resume VM execution on the destination host
  - Fetch memory on demand when accessed
- Reduces downtime over pre-copy
- Can lower performance because of memory access latency
- KVM implementation: Takahiro Hirofuchi & Isaku Yamahata

## Postcopy live migration

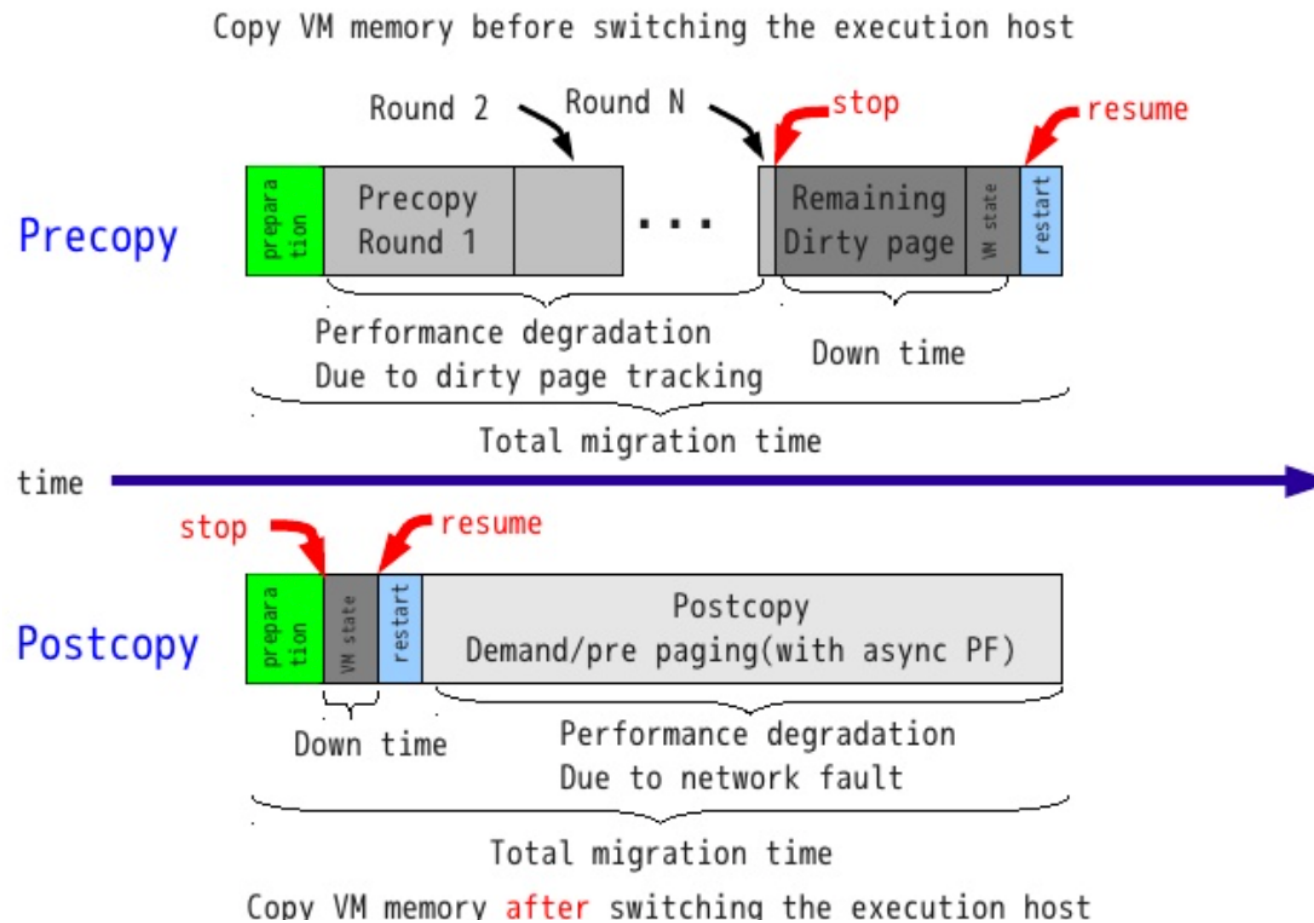
1. Stop VM
2. Copy non-memory VM state to destination
3. Resume VM at destination
4. Copy memory pages on-demand/backgroundly
  - Async PF can be utilized



Copy memory pages

- On-demand(network fault)
- background(precache)

## Total migration/down time



- Metrics to minimize
  - Total data transferred
  - Downtime
  - Total migration time
- Several approaches
  - Data Compression
  - Page Delta Transfer
  - Data Deduplication

- Compress memory pages sent over the network
- Compress zero'd memory pages → available in KVM
  - Interesting for migration of Windows
- Use a compression algorithm (gzip, bzip2, lzo)
  - → KVM supports piping VM state to any executable
- Adaptive memory compression [Jin:2009]



- Memory pages are 4 KB on x86
  - Modify 1 byte in the page → transfer 4 KB
- Delta transfer mechanism:
  - Keep copy of original page
  - Computer differences between original and new page
  - Send diff instead of full content

- VMs can contain identical data in multiple memory pages
- Remove duplicated memory pages
- Fast hash algorithm + full data comparison when match
  - Single-VM [Wood et al., VEE 2011]
  - Multi-VM on same host [Deshpande et al., HPDC 2011]
- Distributed approach for Multi-VM Multi-host
  - [Riteau et al., Euro-Par 2011 ]

# CHAPTER 4.3

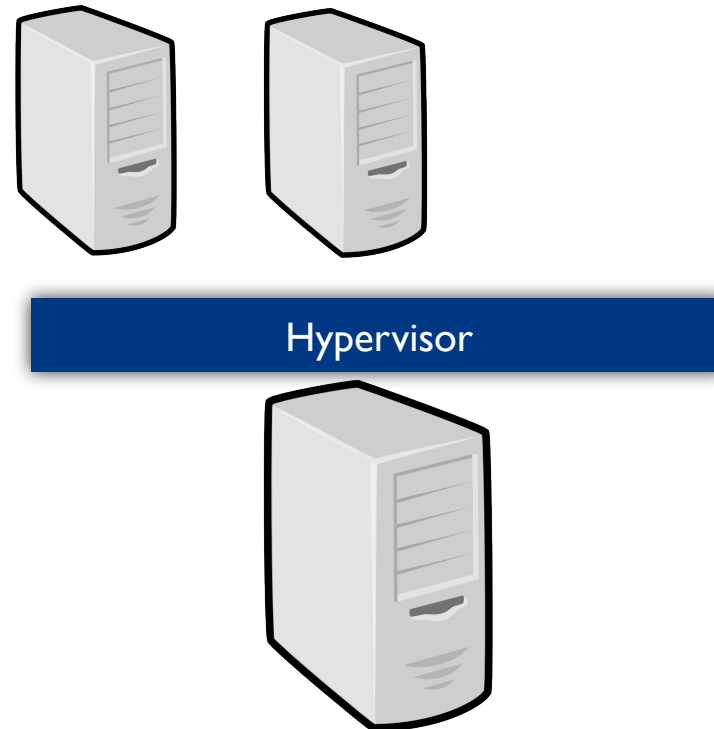
## CONSOLIDATION



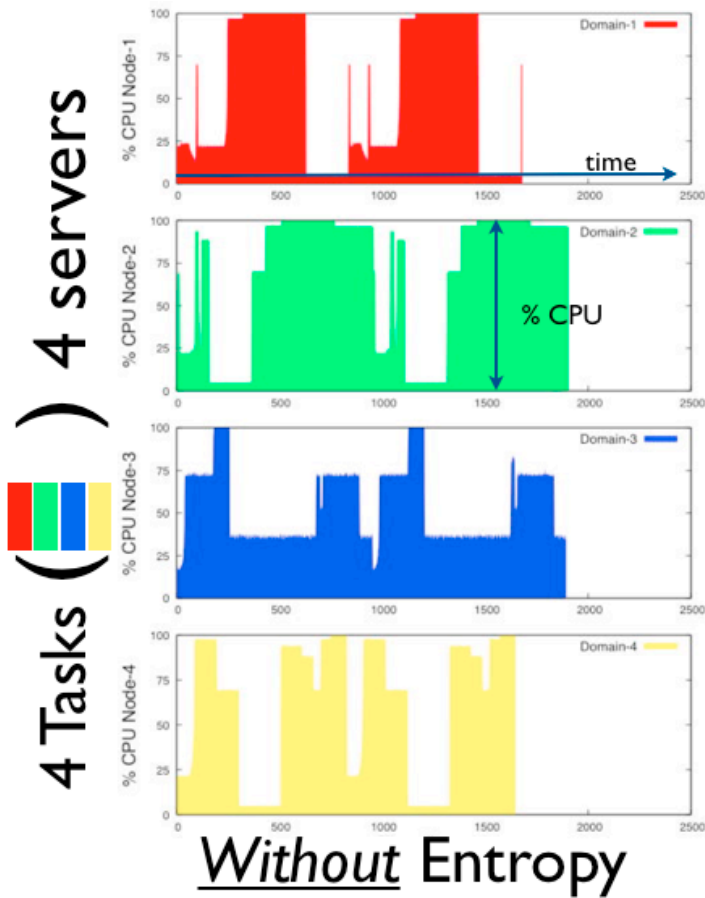
**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

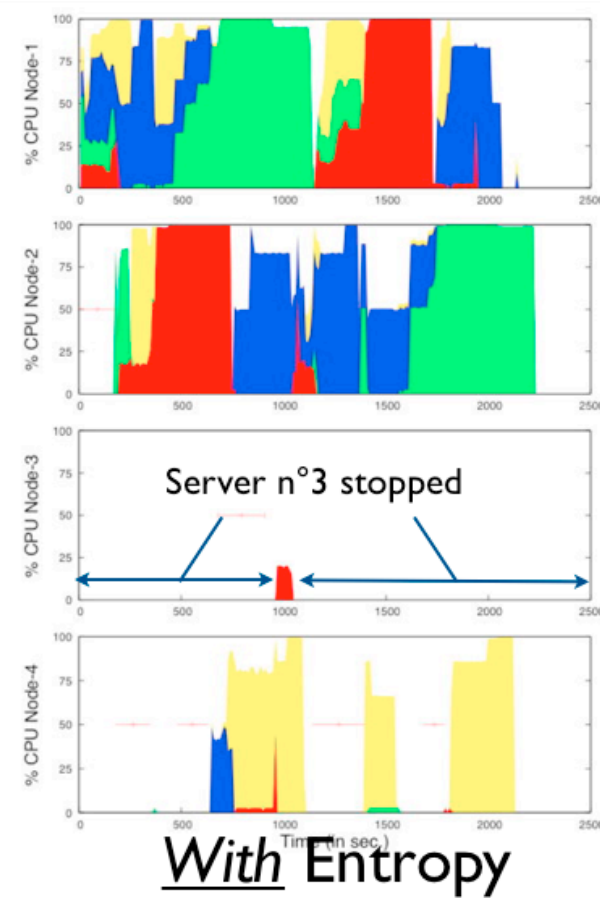
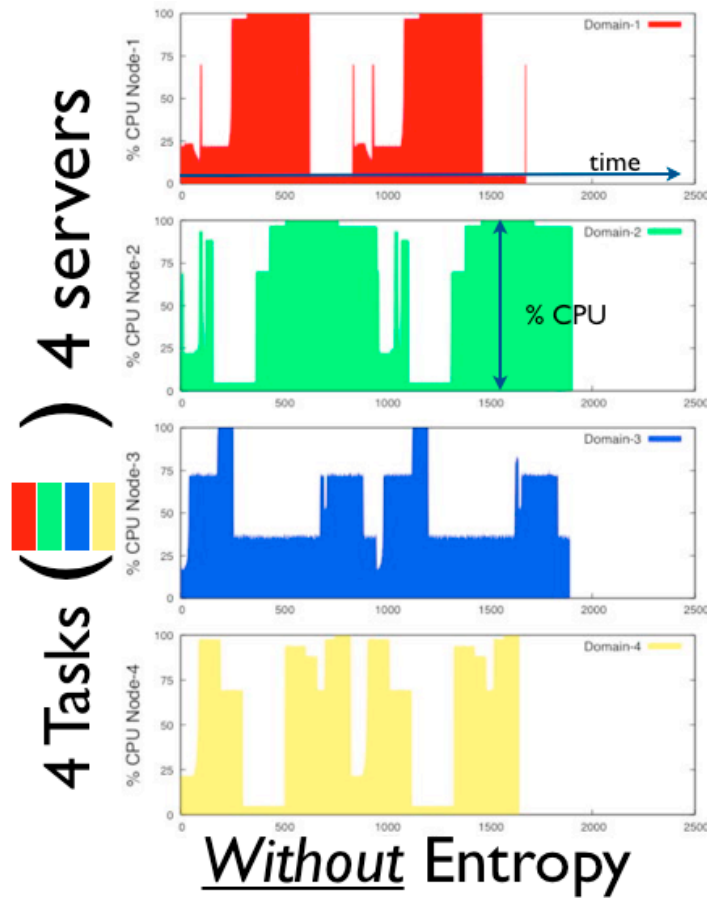
- **Live migration (load-balancing)**
- **High Availability (downtime ~ 60 ms)**

Web EMN Campus



- **Dynamic Consolidation :**
  - The resources are allocated depending on the VM needs
  - VMs are mixed to be hosted on a reduced number of nodes
  - Servers unused can be turned off
  - VMs are remixed when it is necessary





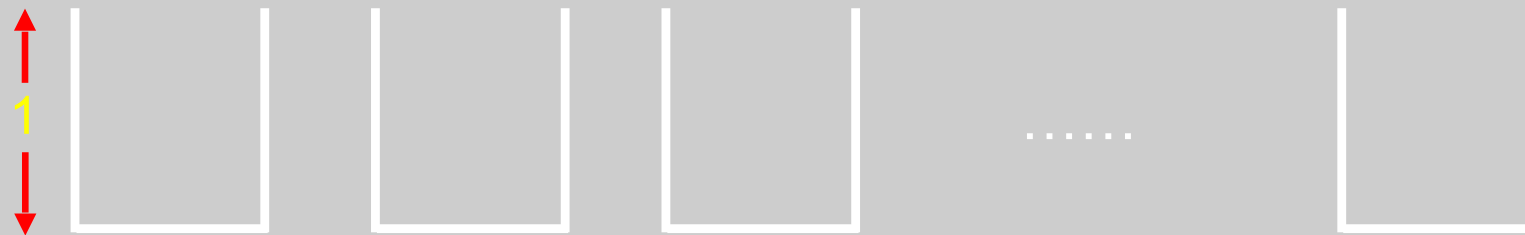
4 Tasks, 3 or 4 Servers  
Consumption is reduced by 25%

# Bin Packing Problem



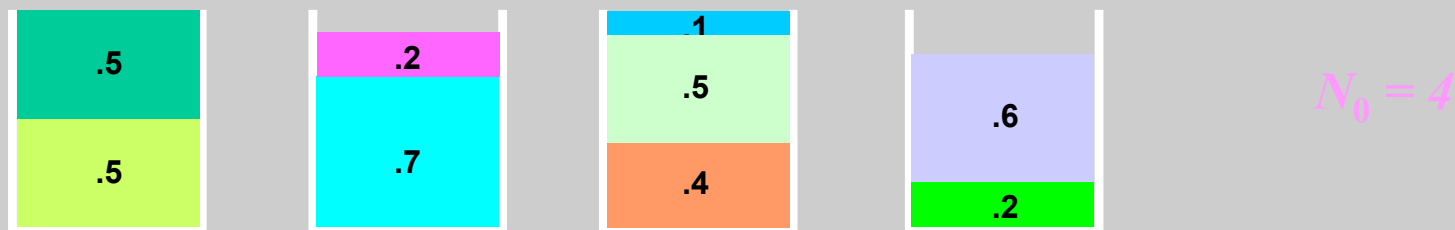


# Bin Packing Problem



.5 .7 .5 .2 .4 .2 .5 .1 .6

## Optimal Packing

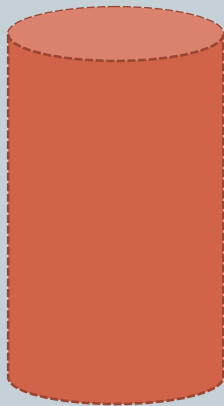


## First fit algorithm

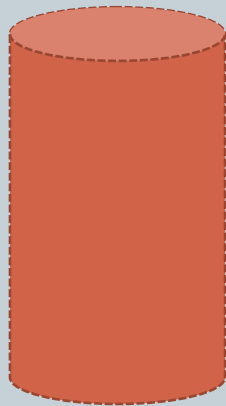
5 7 12 5 3 9 10 6 8 11



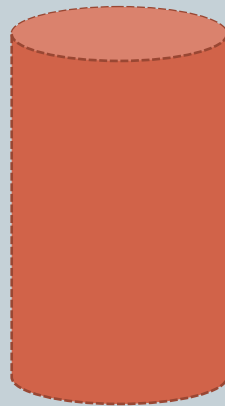
Bin size = 20



Bin 1



Bin 2



Bin 3



Bin 4



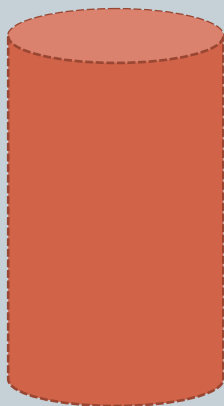
Bin 5

## First fit algorithm

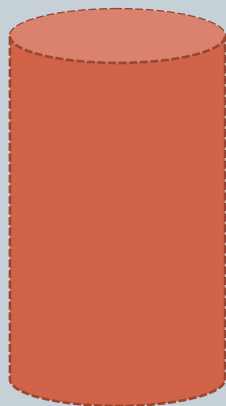
5 7 12 5 3 9 10 6 8 11



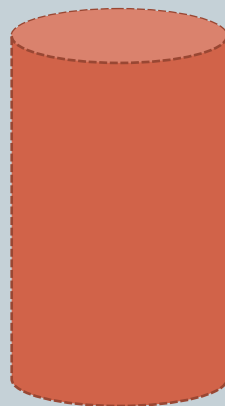
Bin size = 19



Bin 1



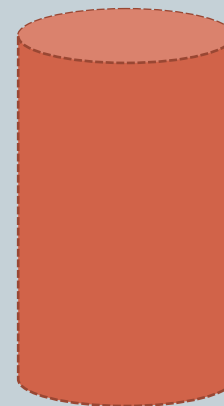
Bin 2



Bin 3



Bin 4



Bin 5

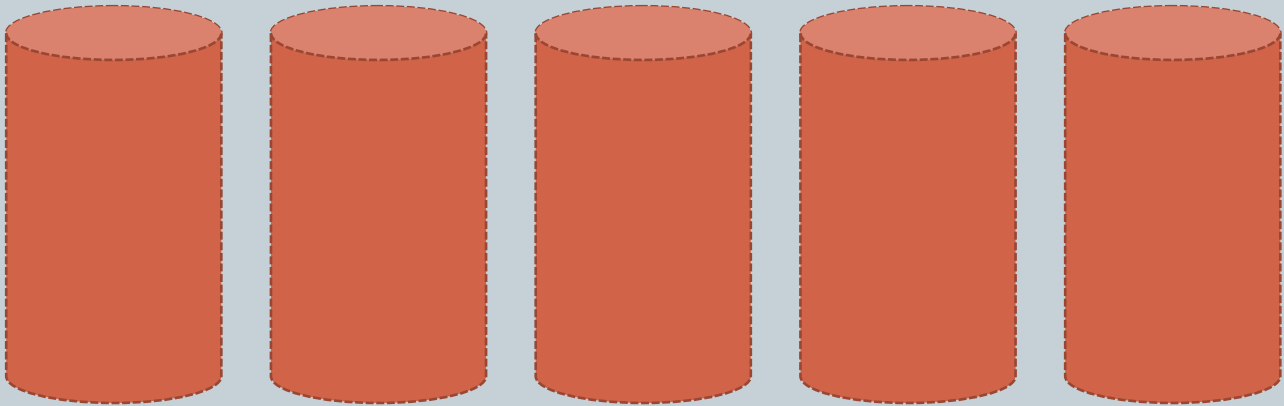
First fit decreasing algorithm



*Sort values in size order, lowest first ...*



Bin size = 20



Bin 1

Bin 2

Bin 3

Bin 4

Bin 5

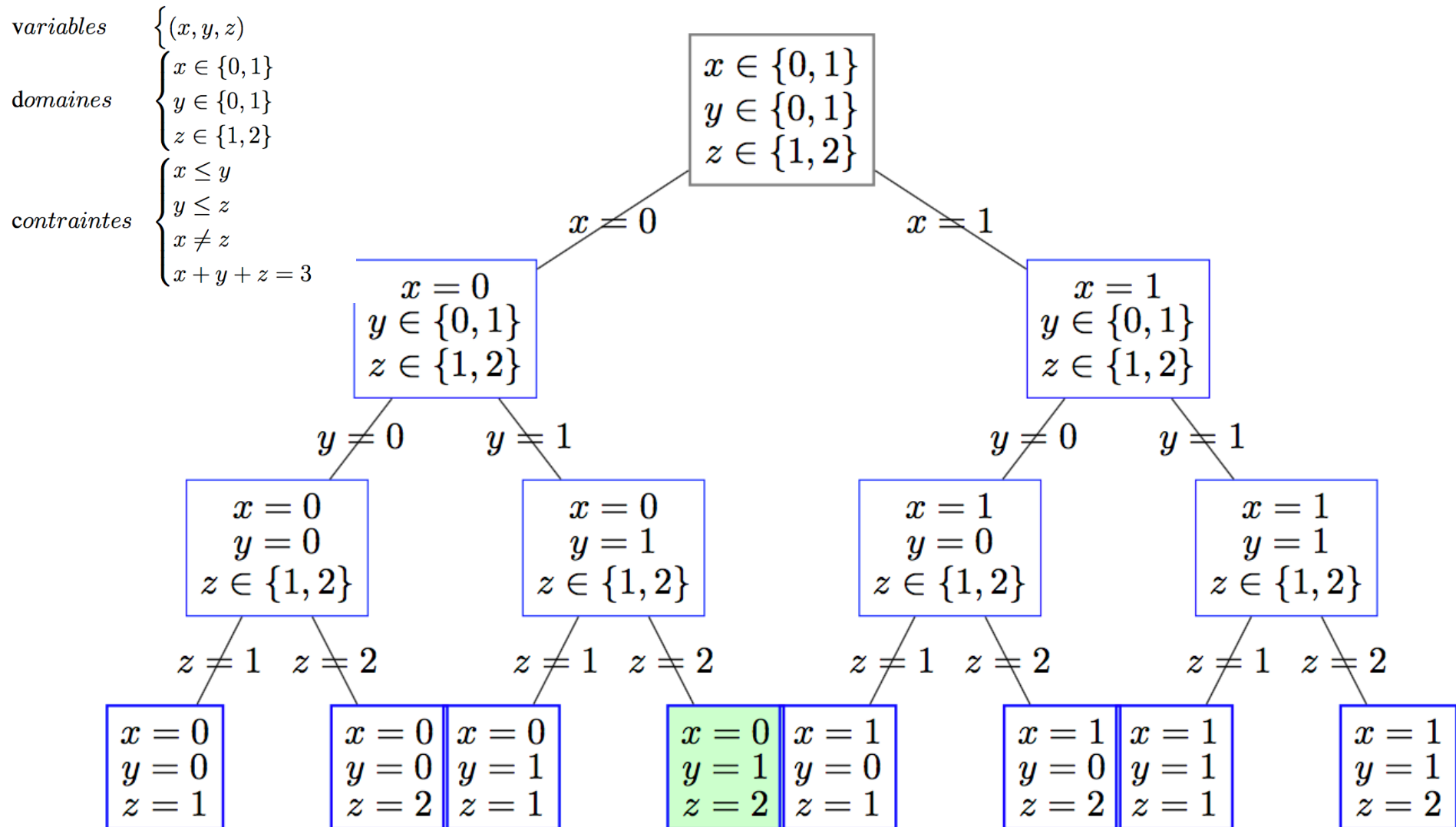
- **Virtual Machine Placement Problem (VMPP) is similar to the multi-dimensional bin packing problem known to be NP-Hard ... [2007-02]**
- **Heuristic methods**
  - Greedy algorithms Ex: EnaCloud [2009-03]
 

Construct a solution by taking local decision without backtrack.  
 First-Fit Decrease (FFD), Best-Fit (BF), Worst-Fit (WF), Next-Fit (NF) ... [1997-01]  
 Pro: Ease to implement, good worst-case complexity  
 Cons: No optimal solution, not really flexible
  - Metaheuristic Ex: Snooze [2012-04]
 

Probabilistic algorithms by searching near optimal solution  
 Genetic, Tabu, Ant colony, Graps ...  
 Pro: Better solution than Greedy algorithms  
 Cons: No optimal solution, not really flexible
- **Exact methods**
  - Mathematical Ex: Entropy [2009-06]
 

Linear or Constraint programming [1986-05]  
 Compute optimal solution  
 Pro: optimal and flexible  
 Cons: Exponential time solving process

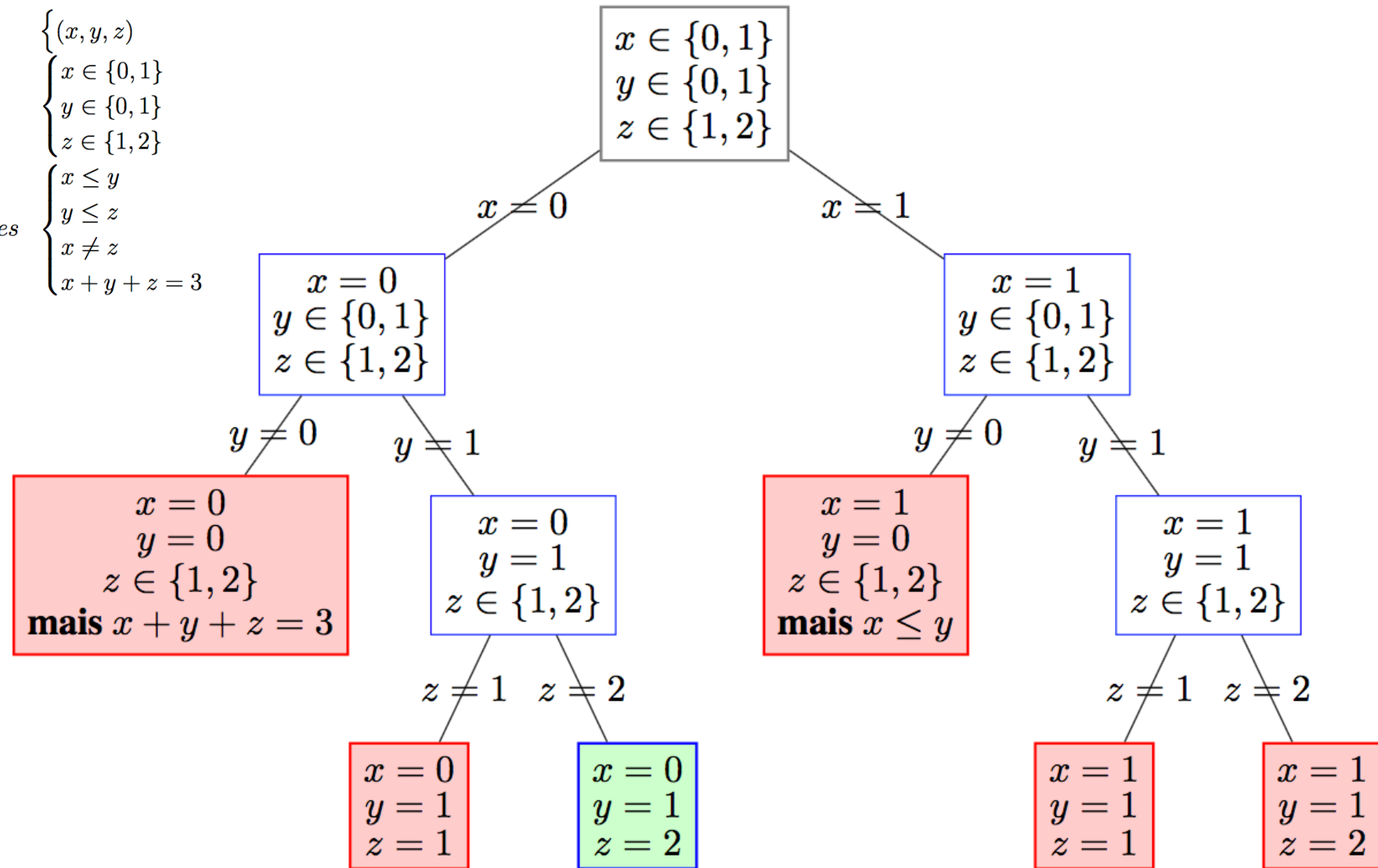
$$\begin{array}{ll} \text{variables} & \{ (x, y, z) \\ \text{domaines} & \left\{ \begin{array}{l} x \in \{0, 1\} \\ y \in \{0, 1\} \\ z \in \{1, 2\} \end{array} \right. \\ \text{contraintes} & \left\{ \begin{array}{l} x \leq y \\ y \leq z \\ x \neq z \\ x + y + z = 3 \end{array} \right. \end{array}$$



variables  $\{(x, y, z)\}$

domaines  $\begin{cases} x \in \{0, 1\} \\ y \in \{0, 1\} \\ z \in \{1, 2\} \end{cases}$

contraintes  $\begin{cases} x \leq y \\ y \leq z \\ x \neq z \\ x + y + z = 3 \end{cases}$

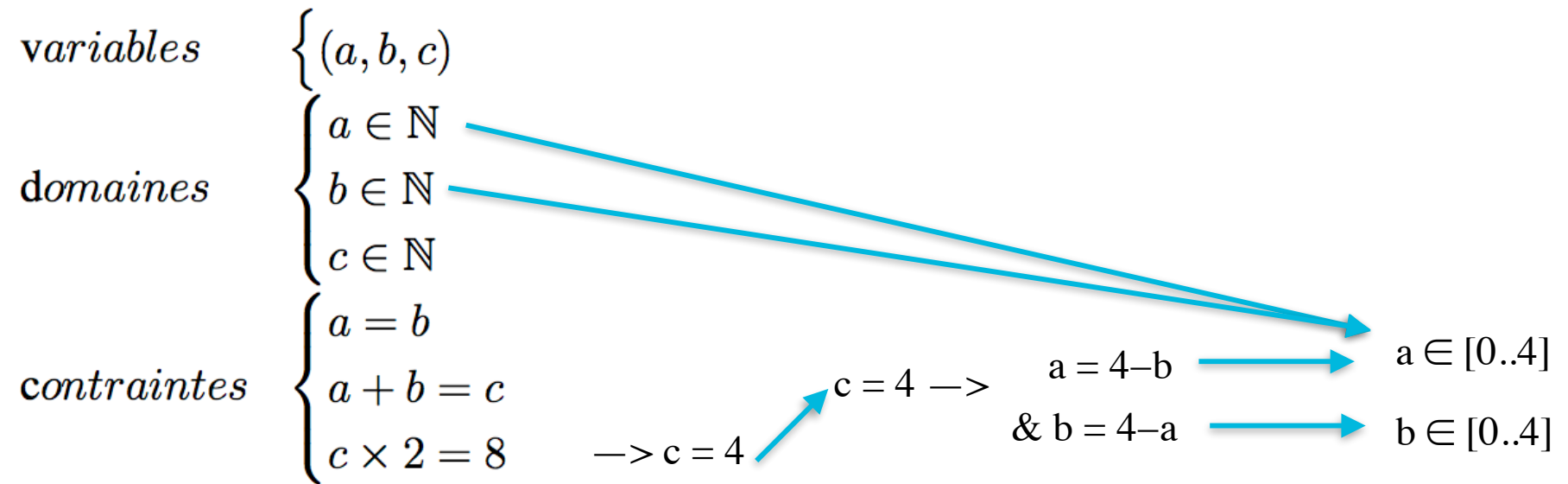


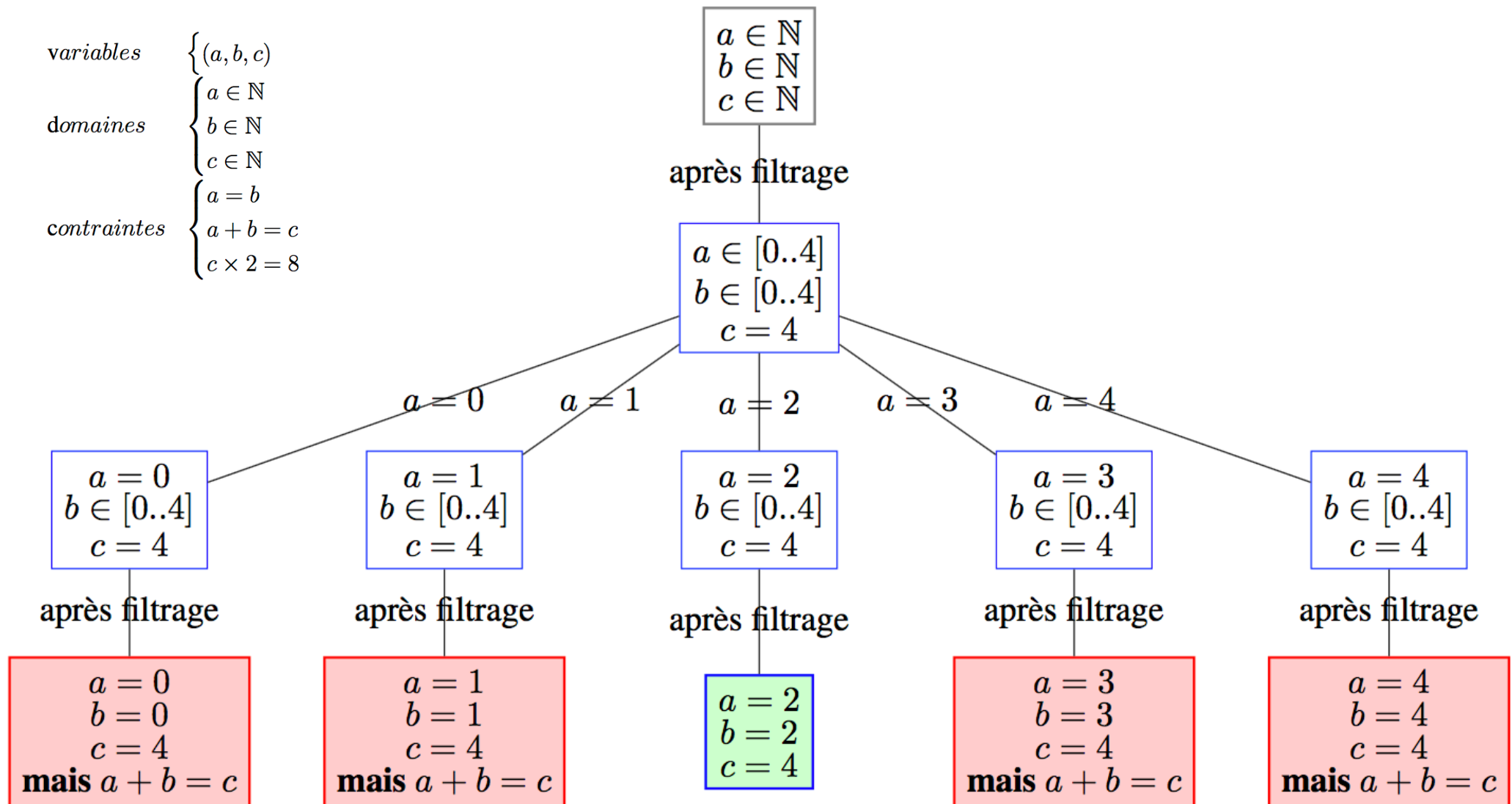


$$\begin{array}{ll} \text{variables} & \{ (a, b, c) \\ \text{domaines} & \left\{ \begin{array}{l} a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \end{array} \right. \\ \text{contraintes} & \left\{ \begin{array}{l} a = b \\ a + b = c \\ c \times 2 = 8 \end{array} \right. \end{array}$$

$$\begin{array}{ll}
 \text{variables} & \{ (a, b, c) \\
 \text{domaines} & \left\{ \begin{array}{l} a \in \mathbb{N} \\ b \in \mathbb{N} \\ c \in \mathbb{N} \end{array} \right. \\
 \text{contraintes} & \left\{ \begin{array}{l} a = b \\ a + b = c \\ c \times 2 = 8 \end{array} \right.
 \end{array}$$

$c = 4 \rightarrow a = 4 - b \ \& \ b = 4 - a$   
 $\rightarrow 2 \times c = 8 \rightarrow c = 4$





- **The approach : constraint programming**
  - generation of a core model
  - placement constraints are translated into "CP constraints"

$$\begin{aligned}\mathcal{X} &= \{x_1, x_2, x_3\} \\ \mathcal{D}(x_i) &= [0, 4], \forall x_i \in \mathcal{X} \\ \mathcal{C} &= \begin{cases} c_1 : x_1 < x_2 \\ c_2 : x_1 + x_2 + x_3 = 4 \\ c_3 : allDifferent(x_1, x_2, x_3) \end{cases}\end{aligned}$$



# CHAPTER 4.4

## VIRTUALIZATION MARKET



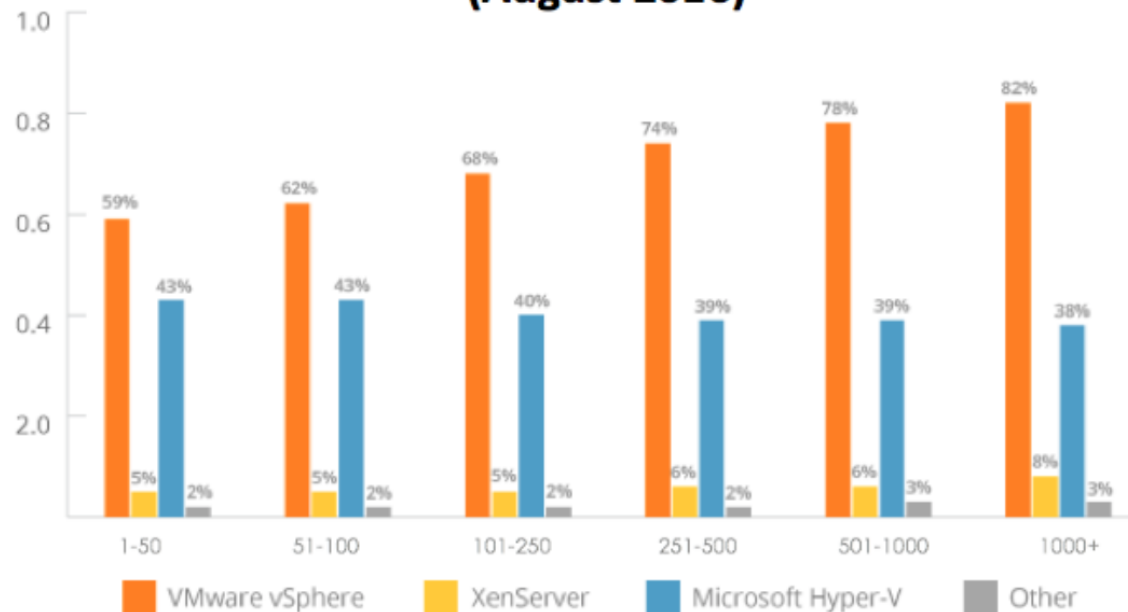
**IMT Atlantique**  
Bretagne-Pays de la Loire  
École Mines-Télécom

Implementation	Technology
<del>KQEMU [9]</del>	<del>Type II full virtualization</del>
KVM [63]	Type II full virtualization
Lguest [72]	Type II para-virtualization
Linux-VServer [48]	OS-level virtualization
LXC [20]	OS-level virtualization
OpenVZ [83]	OS-level virtualization
QEMU [9]	Hardware emulation
Solaris Containers [80]	OS-level virtualization
Sun xVM Server [77]	Type I full virtualization with para-virtualization support
Sun xVM VirtualBox [76]	Type II full virtualization with para-virtualization support
<del>UML [22]</del>	<del>Type II para-virtualization</del>
<del>UMLinux [39]</del>	<del>Type II para-virtualization</del>
VMware ESX & ESXi [86, 88]	Type I full virtualization with para-virtualization support
VMware Workstation [89]	Type II full virtualization with para-virtualization support
Wind River Hypervisor [93]	Type II full virtualization with para-virtualization support
Xen [94]	Type I full- or para-virtualization



Inspection by company size, server virtualization usage at large enterprises heavily favors VMware vSphere while smaller organizations use Hyper-V at a higher rate. For example, 42.6% of the smallest businesses utilize Hyper-V but only 37.9% of the largest companies do. In contrast, 58.6% of the smallest companies use the VMware vSphere ESXi hypervisor, compared to a whopping 82.9% of the biggest companies.

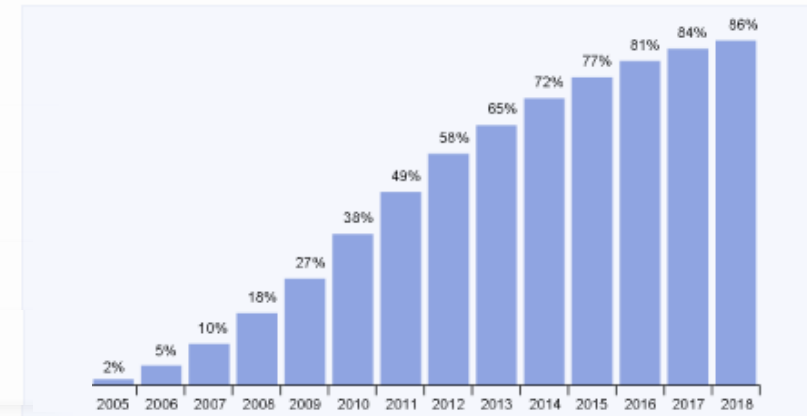
**Server Virtualization Usage Across Company Sizes  
(August 2016)**



- A constant progression
- Q3 2011 [2011-07]
  - virtualization penetration rate: **38.9%**
  - Ratio of virtual machines to physical hosts: **5:1**
  - Primary Hypervisor Usage for Server virtualisation: **ESX 67,5%**

	Avg	UK	FR	DE	US
VMware	67.6	79	65	61	66.5
Citrix	14.4	12	15	20	12.5
Hyper-V	16.4	8	17	16	20.5
Other	1.6	1	3	3	0.5

Figure 1. Percentage of x86-Architecture Workloads Running in VMs



Source: Gartner, March 2011

Research: Virtual Machines Will Slow in the Enterprise, Grow in the Cloud

Gartner March 2011