Cloud & Virtual Machines

High Performance Computing
(Computació d'Altes Prestacions)

Josep Lluís Berral-García – josep.ll.berral@upc.edu
Jordi Torres Viñals – jordi.torres@upc.edu

BY NC SA

Introduction

"Real experiments and use cases will not fit in your laptop…

…better borrow infinite resources from the Cloud"

Session Objectives

- Explain when/why we need to execute AI in scalable systems
- Explain how Virtual Machines work & how to use them
- Explain how image repositories help on exporting our AI software

Off-Loading and Scaling in the Cloud

CLOUD COMPUTING

Cloud Computing

E.g., a data-center from Google

- Computing resources "as a Service"
  - Borrow machines, platforms, processes
  - Pay per resource/time

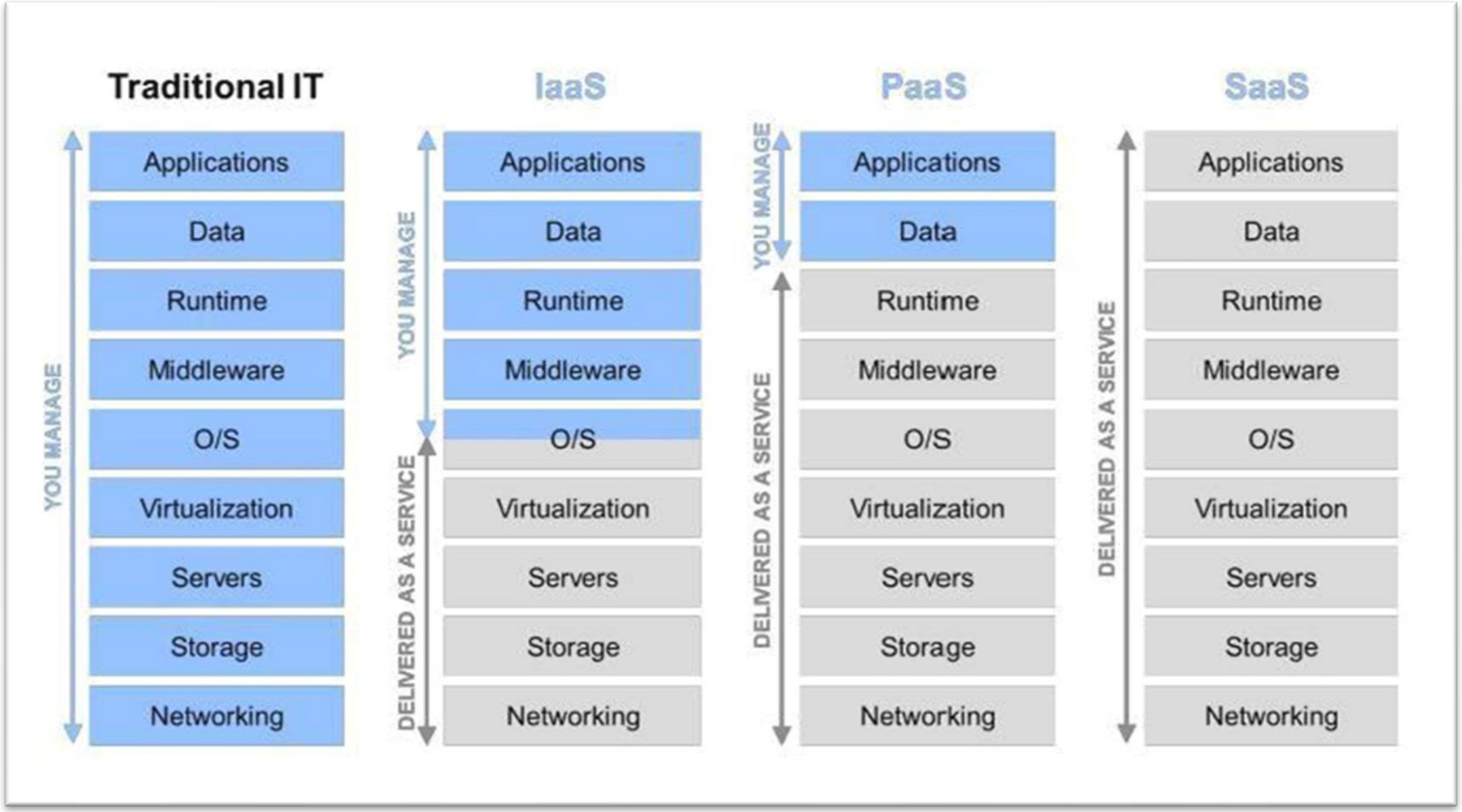
Cloud Computing

- Cloud Computing:
  - Computing resources "as a Service"
  - Where are the resources?
    - No need to know…

- Large Data-Centers
  - Users "borrow" resources
    - "Machines" (actually, virtual machines)
    - "Platforms", like (e.g.) a Pytorch cluster
    - "Services", like (e.g.) website or database space
  - Users pay Providers per "resource/time"

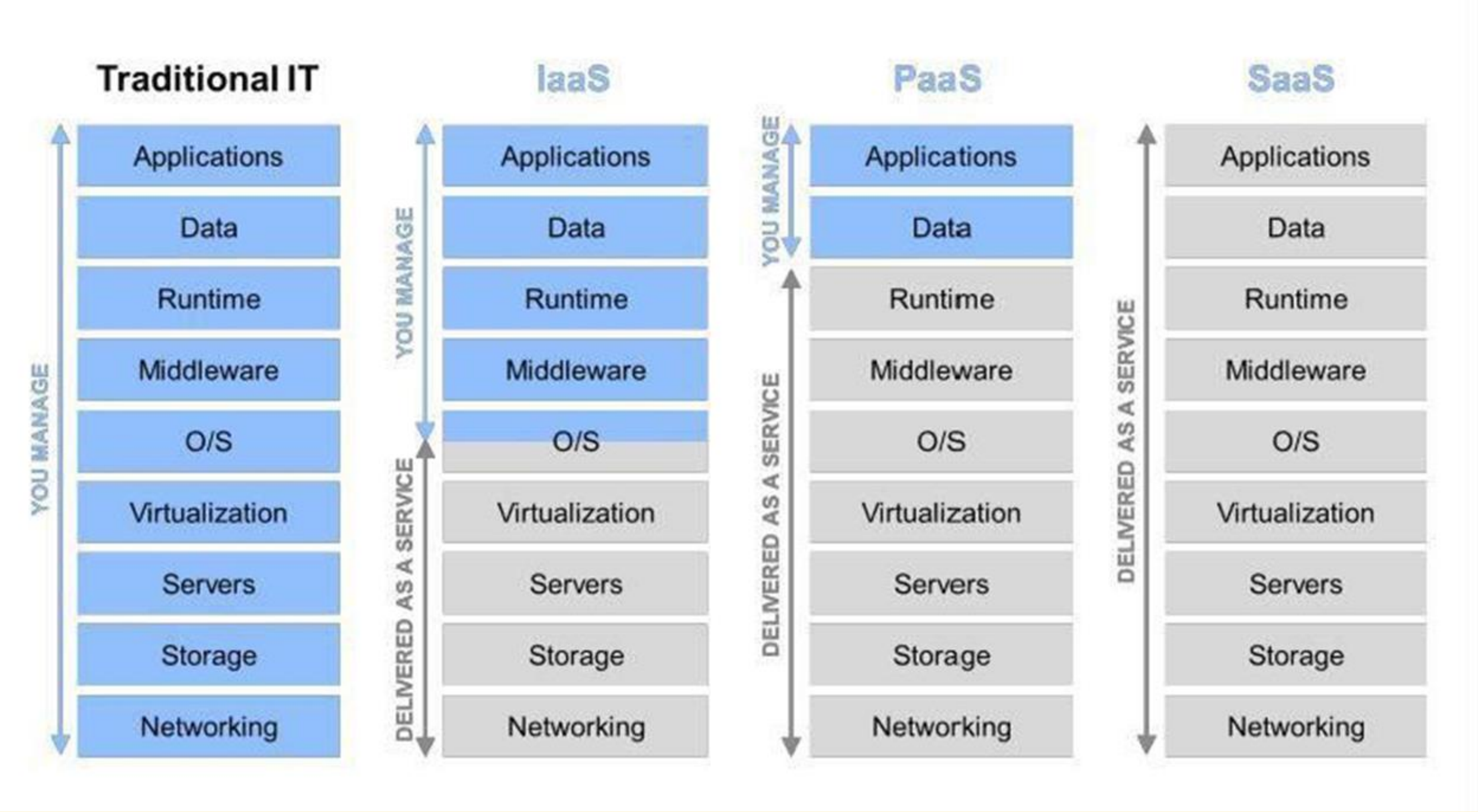
Everything as a Service

- Hardware/Software Stack



Everything as a Service

- Hardware/Software Stack



Examples

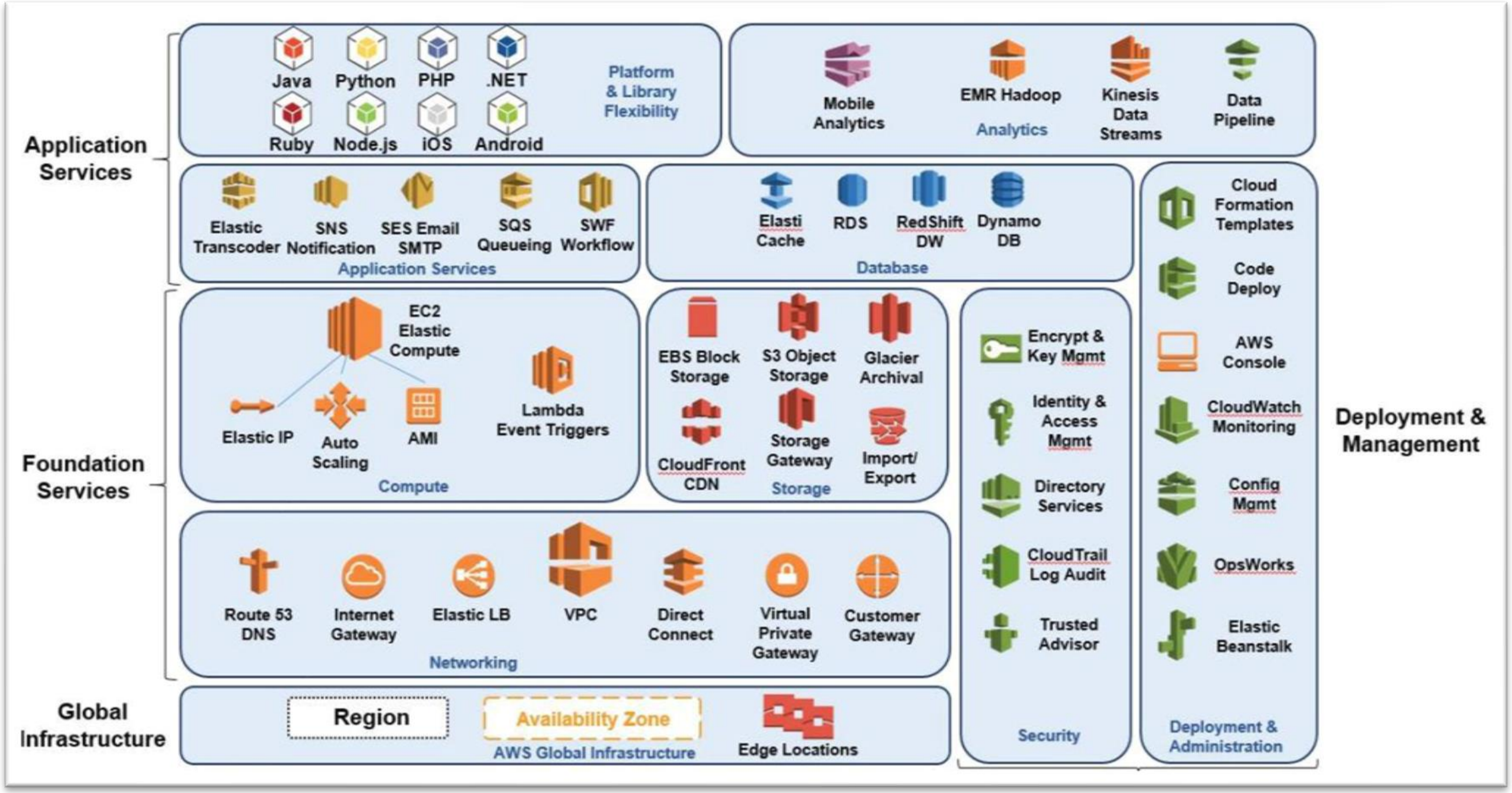
- Apps, ML Scripts, Webservices, …
- Datasets, Websites, …
- Spark, TensorFlow, PyTorch, MySQL, OpenMP, MPI, Apache, Nginx, PHP, ...
- Ubuntu, Windows, ...
- OpenStack, OpenShift, KVM, VBox, ...
- Hardware (CPUs, GPUs, TPUs, Disks, Network Interfaces, ...)

Everything as a Service

- Infrastructure as a Service
  - The provider provides the virtual machines and networks
  - We configure the architecture of our computing cluster
  - We install the operating system and our software, upload the data and process it
- Platform as a Service
  - The provider provides the virtual machines and middleware software
  - We install the extra software, upload the data and process it
- Service as a Service
  - The provider provides the virtual machines, middleware and all software
  - We upload the data and process it

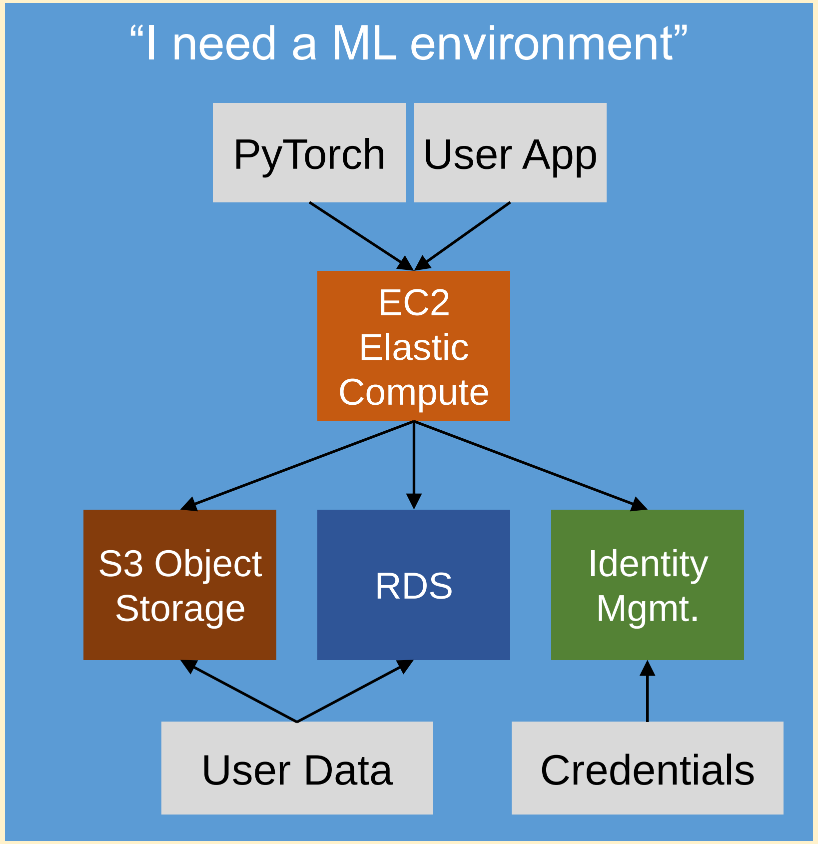
Example of Commercial Services

- E.g., Amazon Web-Services



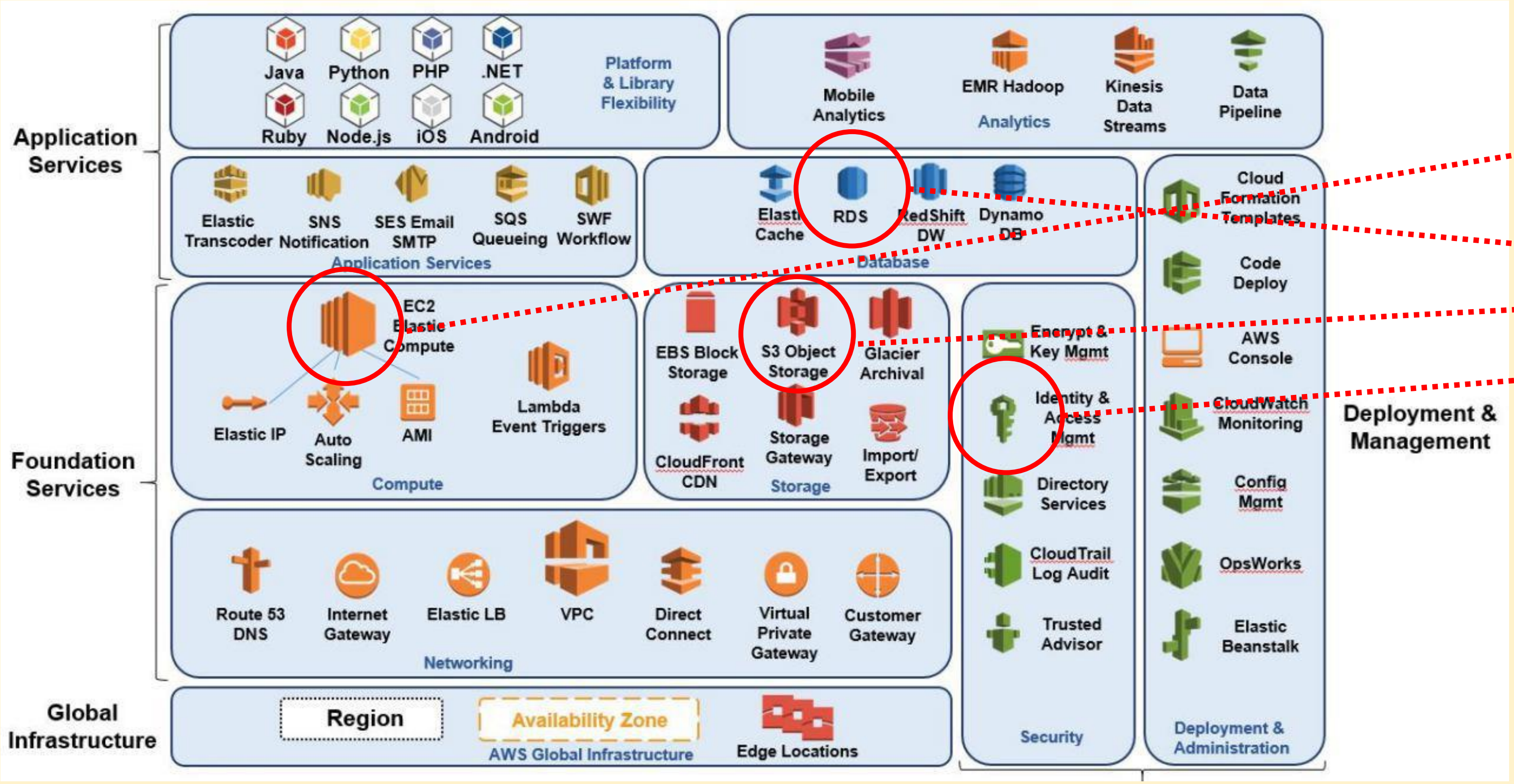
Example of Commercial Services

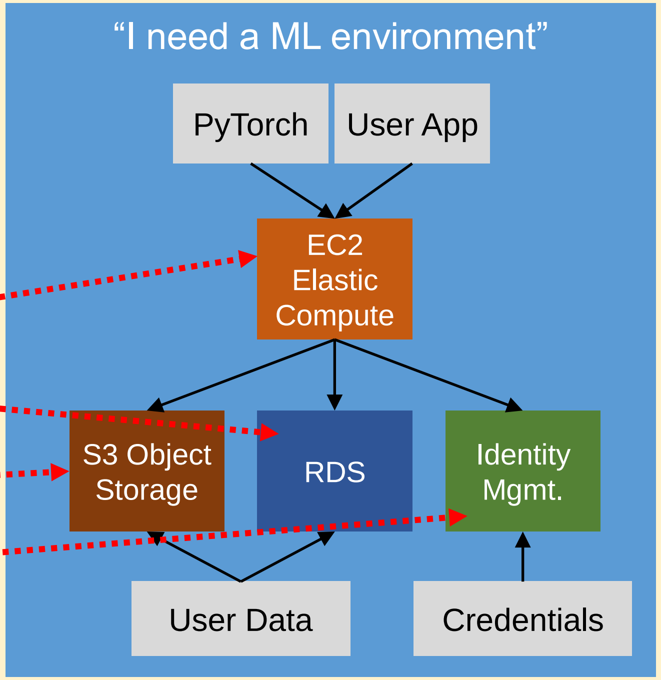
- Suppose we want to create an ML environment
  - PyTorch + User Application
- Infrastructure in the Cloud (e.g., Amazon)
  - Computing machines: EC2 Elastic Computing
  - Data storage: S3 Object Storage
  - Data persistence: RDS Relational Data Base
  - Security: Identity Management
- Monthly Bill:
  - EC2: €/h x VMs uptime
  - RDS: €/GB x used space
  - IdM: €/acc x accesses
  - S3: €/GB x used space



Example of Commercial Services

- E.g., Amazon Web-Services

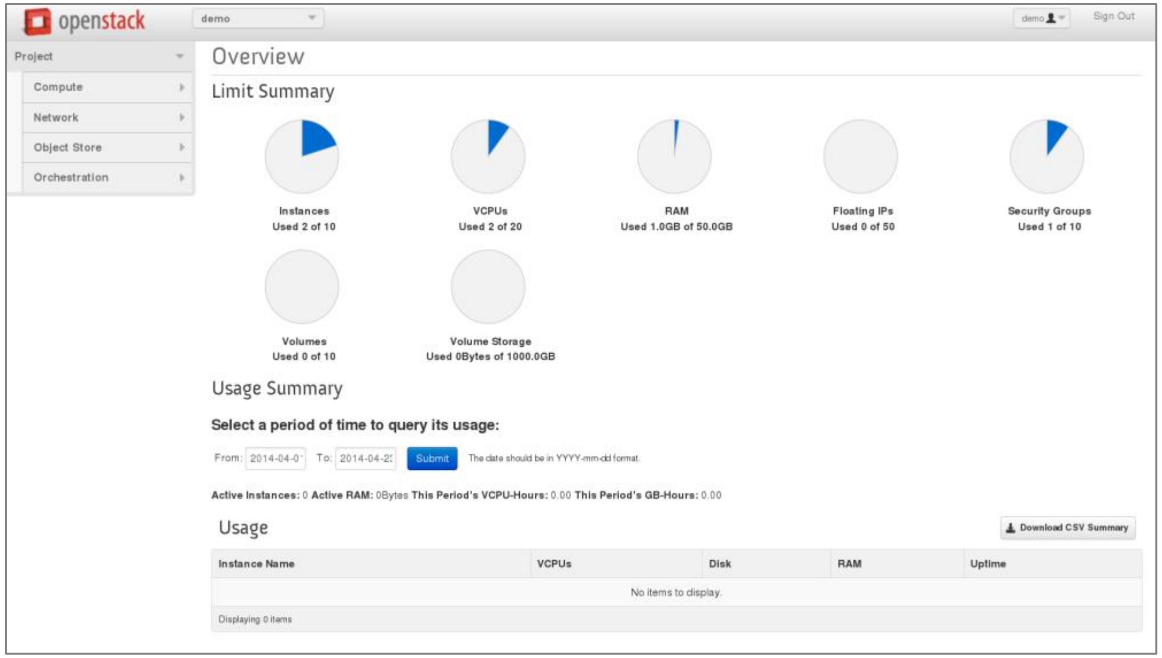




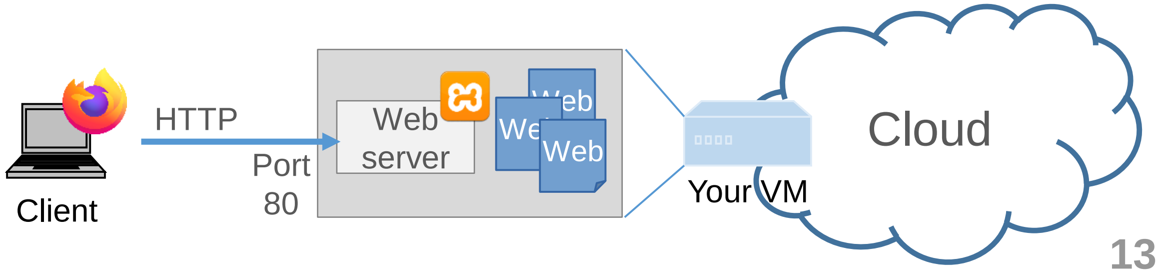
Monthly Bill:
```
–  EC2:  €/h x VMs uptime
–  RDS:  €/GB x used space
–  IdM:  €/acc x accesses
–  S3:   €/GB x used space
```

Accessing the Cloud

- Access to the Cloud Services (examples)
  - As developer/manager
    - Management Portal, CLI tools
  - As developer/user
    - REST APIs, CLI tools
  - As client
    - REST API from your apps or services

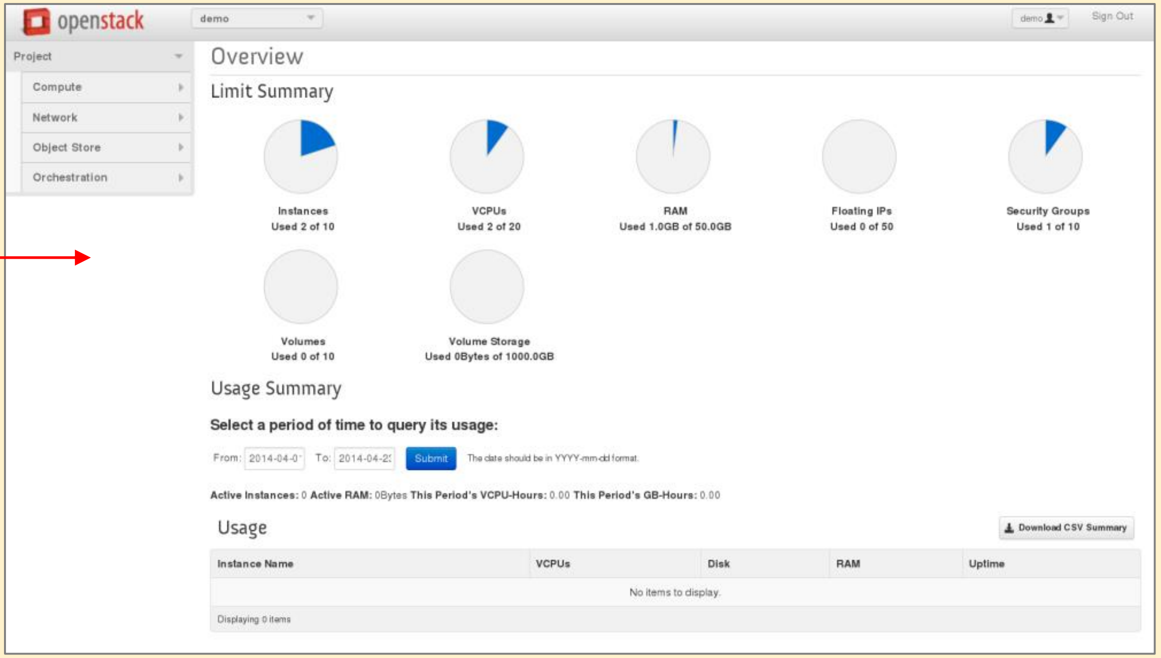



```
$ ssh username@cloud-login.xxxxx.com -p 2222 -IdentityFile ~/.ssh/id_rsa
```

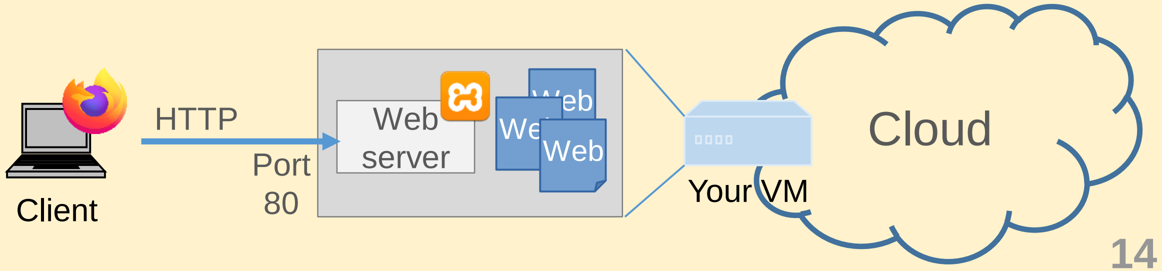


Accessing the Cloud

- Access to the Cloud Services (examples)
  - As developer/manager
    - Management Portal (visual)
    - Console tools, allowing to check our infrastructure, services and applications running (console)
  - As developer/user
    - REST APIs, indicating how our application must communicate with the Cloud (protocol)
    - Console tools, e.g. SSH access (console)
  - As client
    - REST API, e.g. web protocol, on how the clients communicate with your apps or services (protocol)




```
$ ssh username@cloud-login.xxxxx.com -p 2222 -IdentityFile ~/.ssh/id_rsa
```



Accessing the Cloud

- Also examples
  - Access to "boada", "cromai", "marenostrum"…
    - Console through "SSH"
  - Access to Amazon or OpenStack
    - Console application "mc"
    - Web portal (visual management)
  - Access to Services
    - Web access through browser
    - Console access through "telnet", "wget", "curl"

Resources Placement

- Different Placements in the World
  - Geographically placed 'Zones'

Americas

Berkeley County, South Carolina
Council Bluffs, Iowa
Douglas County, Georgia
Jackson County, Alabama
Lenoir, North Carolina
Mayes County, Oklahoma
Montgomery County, Tennessee
Quilicura, Chile
The Dalles, Oregon

Asia

Changhua County, Taiwan
Singapore

Europe

Dublin, Ireland
Eemshaven, Netherlands
Fredericia, Denmark
Hamina, Finland
St Ghislain, Belgium

* Google's Data-Centers in 2018

Resources Placement

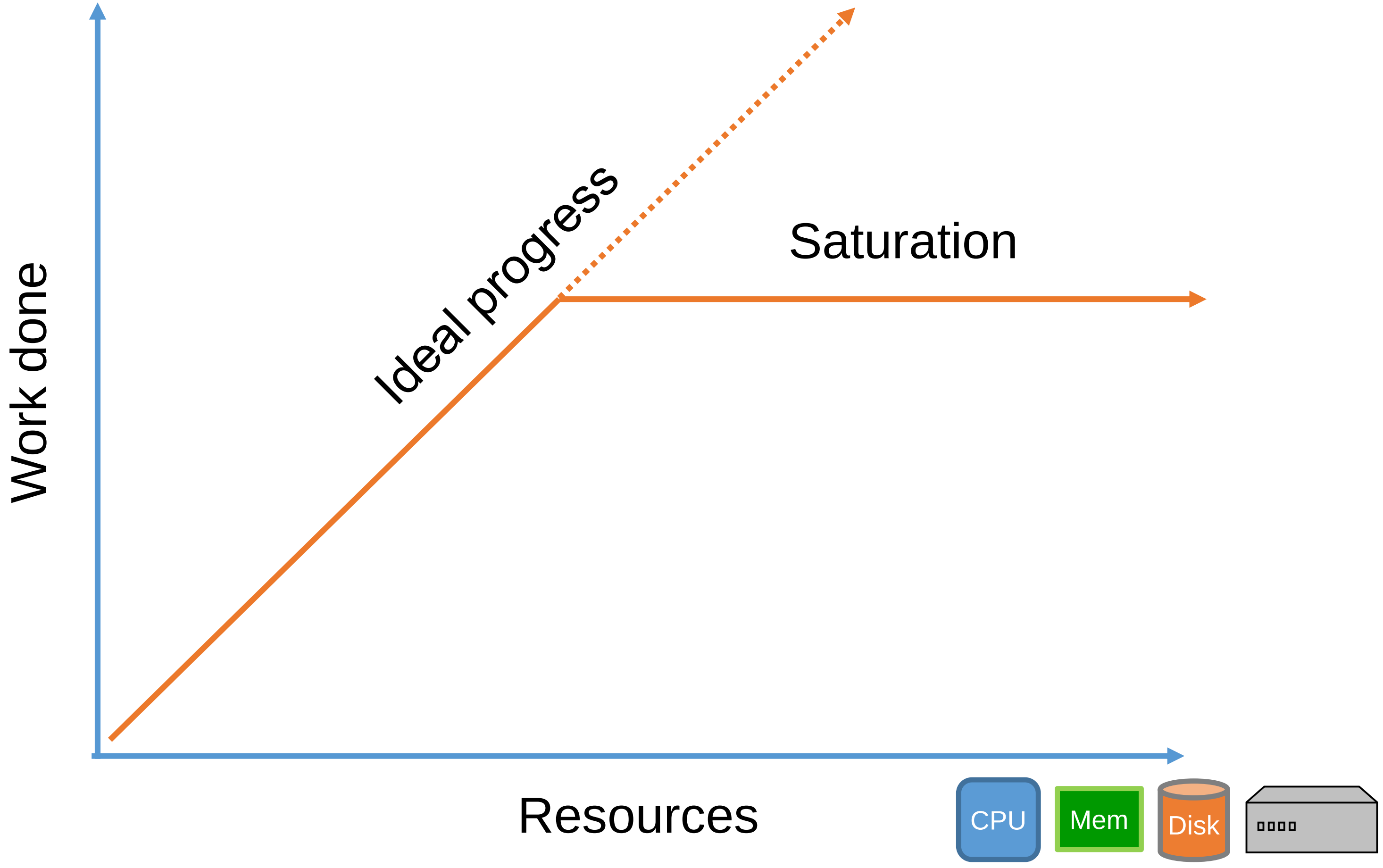
- Different Placements in the World

  – 'Areas' or 'Zones'
    - Providers divide the world in "zones" (countries or regions)
    - Each zone might be under different legal regulations
    - Also, each zone is geographically in a place, that can be near or far from ourselves or our clients.

  – Beware! Your data and applications might be forbidden to place outside your country or in certain countries

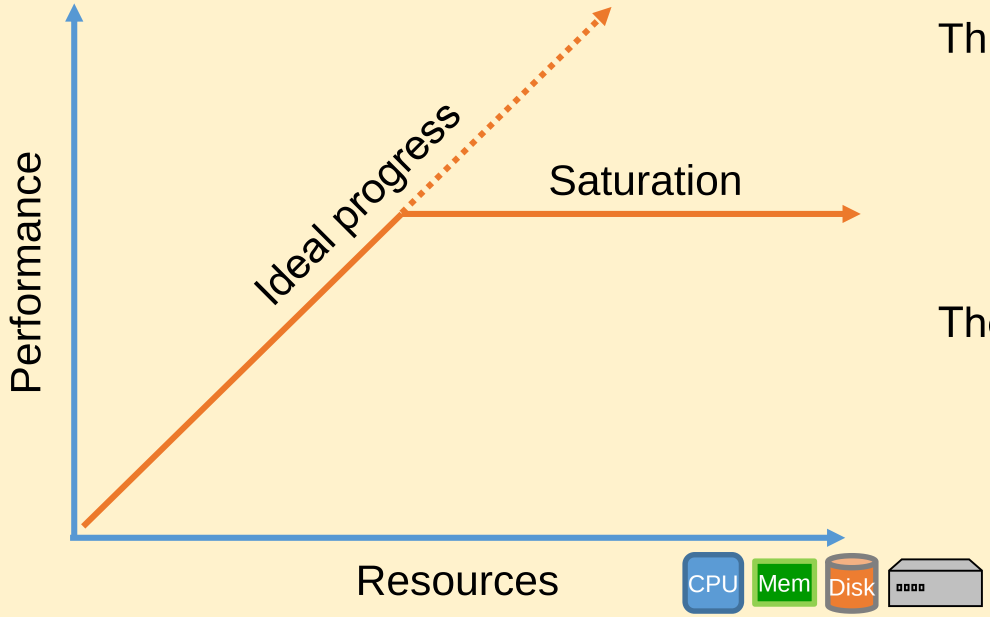
Scalability

- Performance



Scalability

- Performance
  - E.g., throughput: avg output per time-unit
  - E.g., runtime: avg execution time per job
  - E.g., experiments finished per time-unit
  - E.g., data-sets processed per time-unit
  - E.g., data points trained per time-unit



This works for

- Jobs, HPC and HPDA
- Services
- App instances

The Cloud → Scale INFINITE…

- Check on distribution overheads
- Check on architectures and pipelines (bottlenecks)
- Scale up (more resources per job) and Scale out (more jobs & machines)

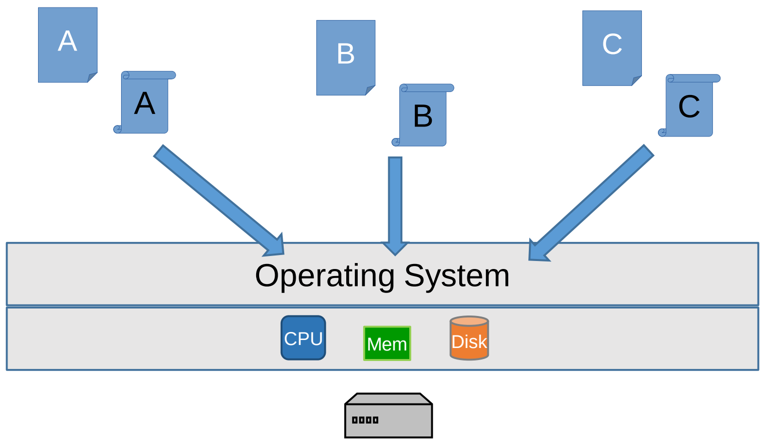
Workload Isolation and Migration

VIRTUALIZATION

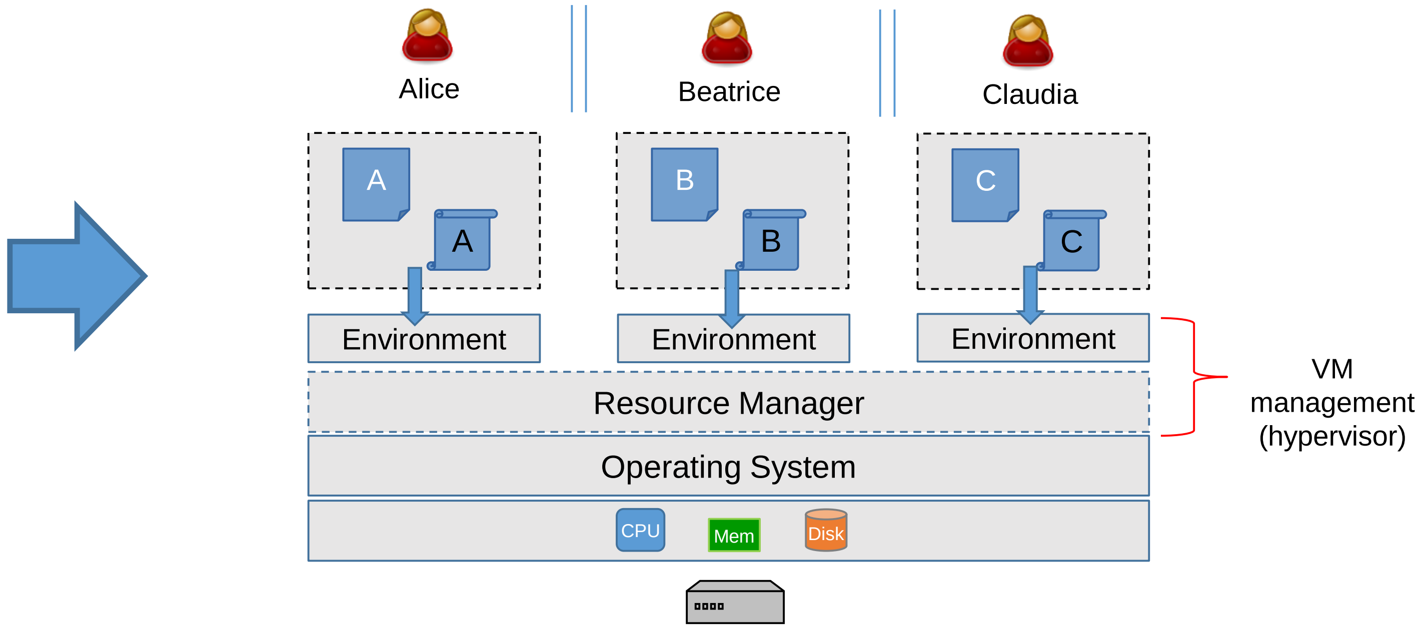
Environment Isolation

- Isolation of Executions

Physical / Native Machine



Virtualized Environment



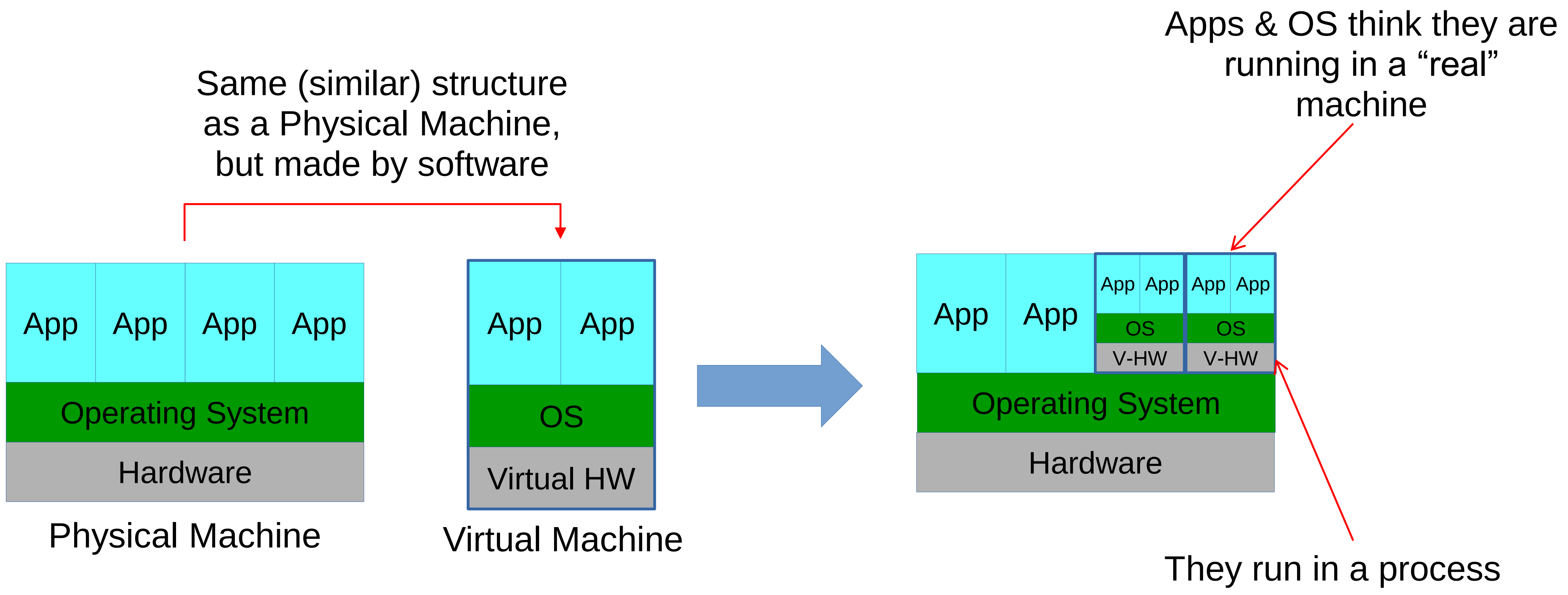
Environment Isolation

- Physical / Native Machine
  - All processes share environment:
    - Resources (CPU, Mem, I/O…)
    - Processes can interact

- Virtualized Environment
  - Each user has its own environment:
    - Has a quota on Resources (CPU, Mem, Disk…)
    - Processes across **environments can't interact**

Virtualization

- A "machine inside your machine"

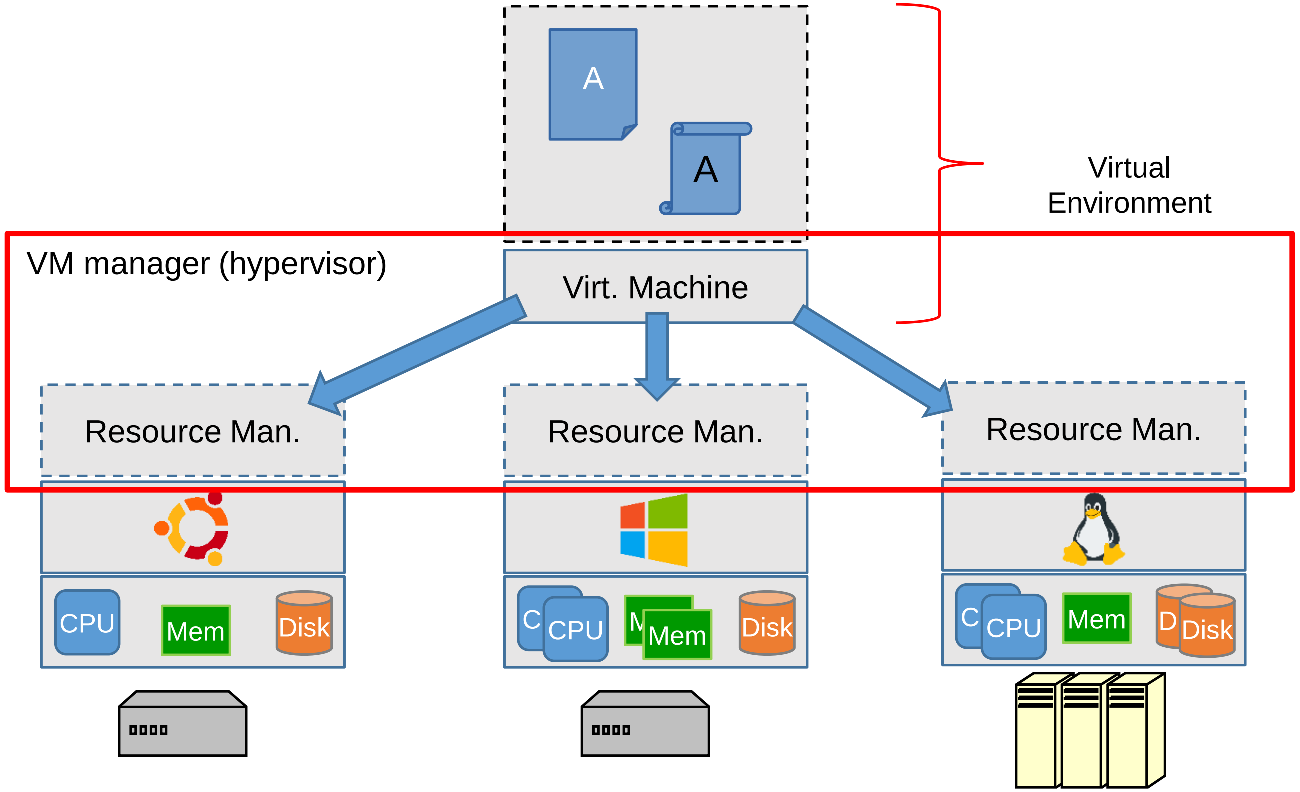


Virtualization

- A "machine inside your machine"
  - VM → Partial or total emulation of a "real machine" as a process
    - Each process is granted a quota of resources (physical or virtual)
    - Overhead: executing additional layers (HW emulation and Guest OS)
    - OS and Hypervisor prevent unauthorized memory share and communications
- Security
  - Users in VMs must be uncapable to discover other users/apps
  - Trust between Host-OS & Guest-OS

Virtualization

- Virtualization:
  - Resources are "virtualized"
  - Independent of the underlying infrastructure



Virtualization

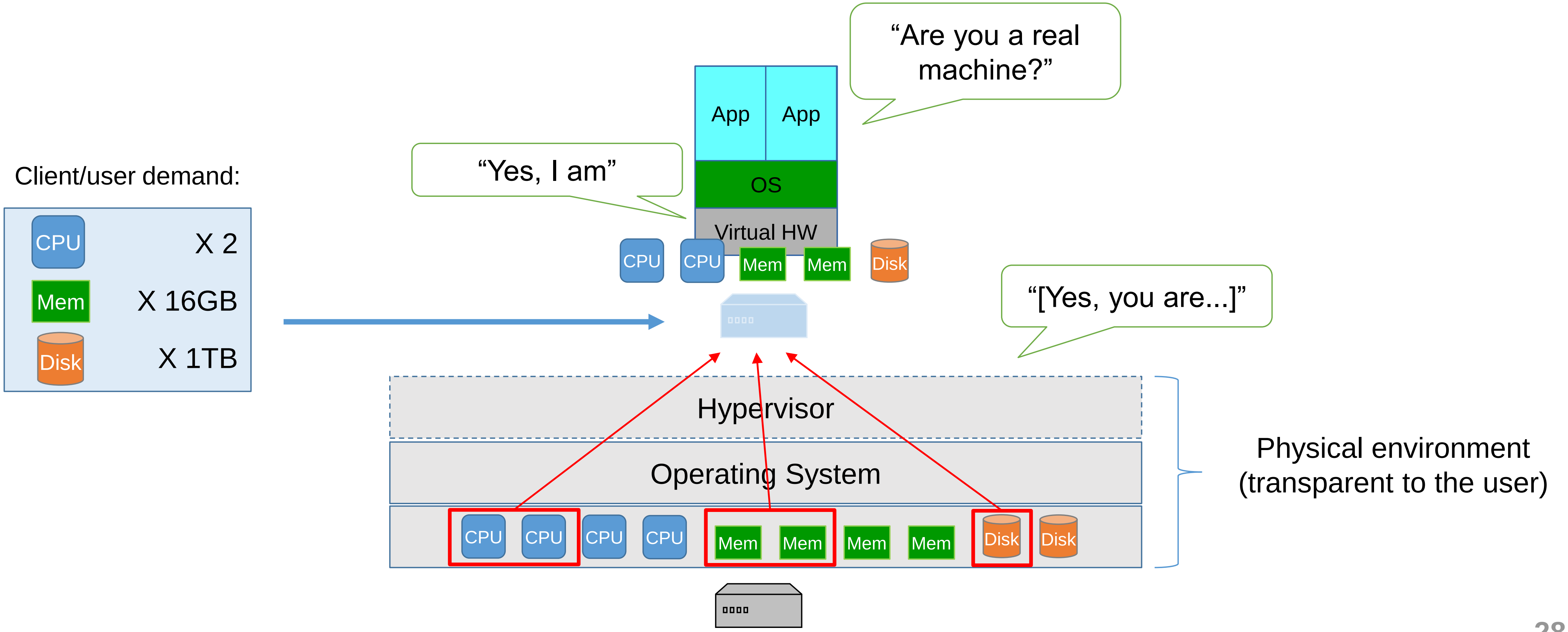
- Virtualization:
  - Resources (e.g. CPU/Mem/Disk…) are "virtual"
    - Transforms virtual operations into host-machine operations
    - … or passes operations to allowed-access HW
  - Independent of the underlying infrastructure
    - Full virtualization (including HW) → All is emulated, full-stack
    - Para-virtualization (leverage HW) → Some parts are emulated, some parts access to real HW

Para-Virtualization

- Examples:
  - QEMU → Hyper-visor emulates full HW-SW stack
    - Allows virtualization on heterogeneous platforms
      - E.g., PPC/RISC-V/... on x86/64 systems
  - KVM, VBox, Xen, ... → Leverage existing HW
    - Use of VT-x technologies (specialized registers/components for VM processes)
      - Specific driver modules from Host OS and Guest OS
      - Special care on security and isolation through real HW/OS

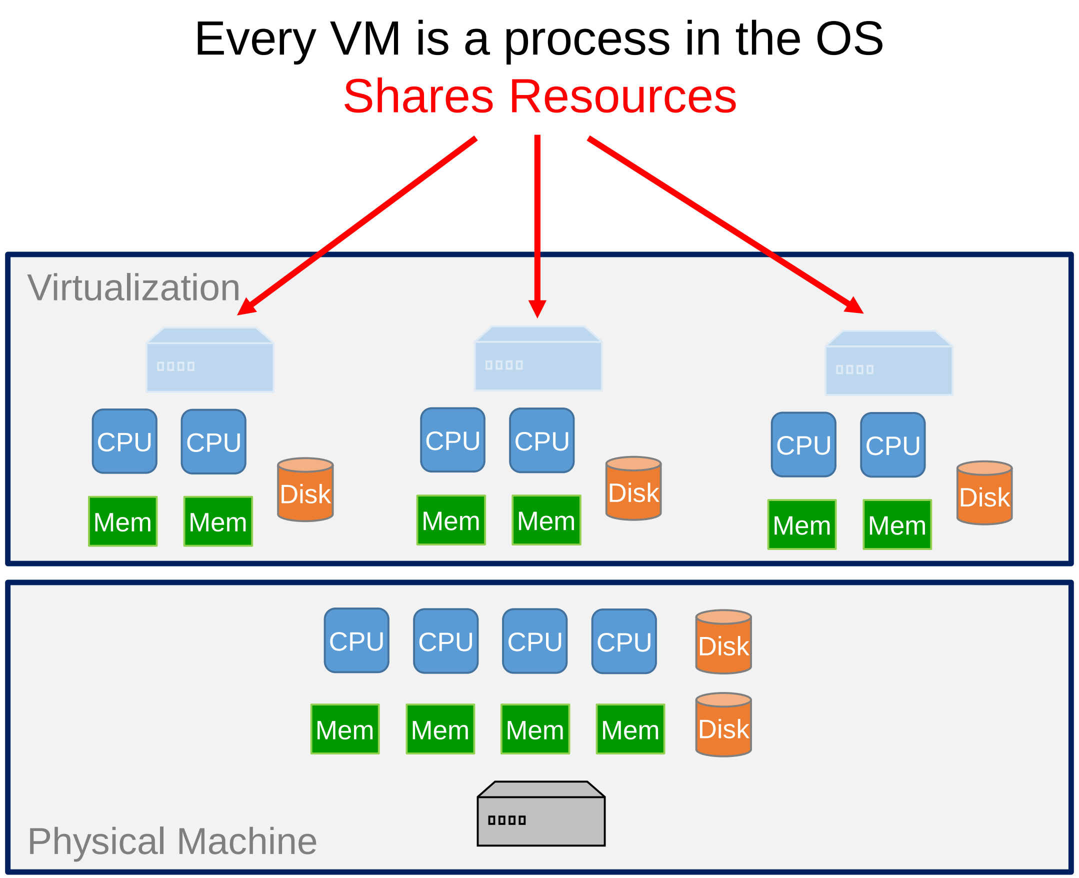
Tailoring Resources

- Dimension your Environment



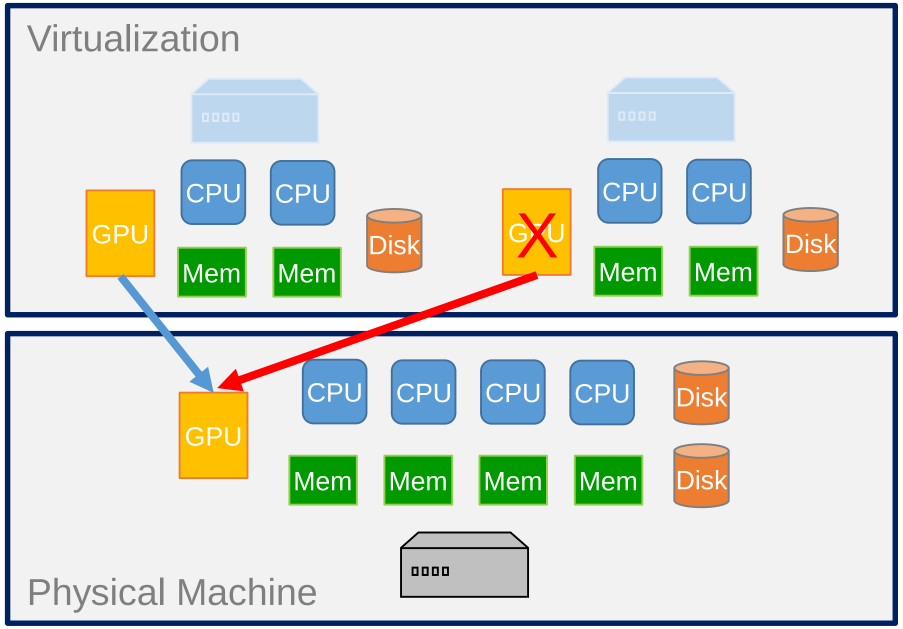
Sharing the HW

- Beware: Such resources are NOT exclusive!
- Strategy of "Consolidation"
  - Packing VMs over resources
  - Sharing resources



Sharing the HW

- Sharing Accelerators (GPUs and TPUs)
  - Some devices cannot be "virtualized"
  - Resources Collision → Competition / Eviction

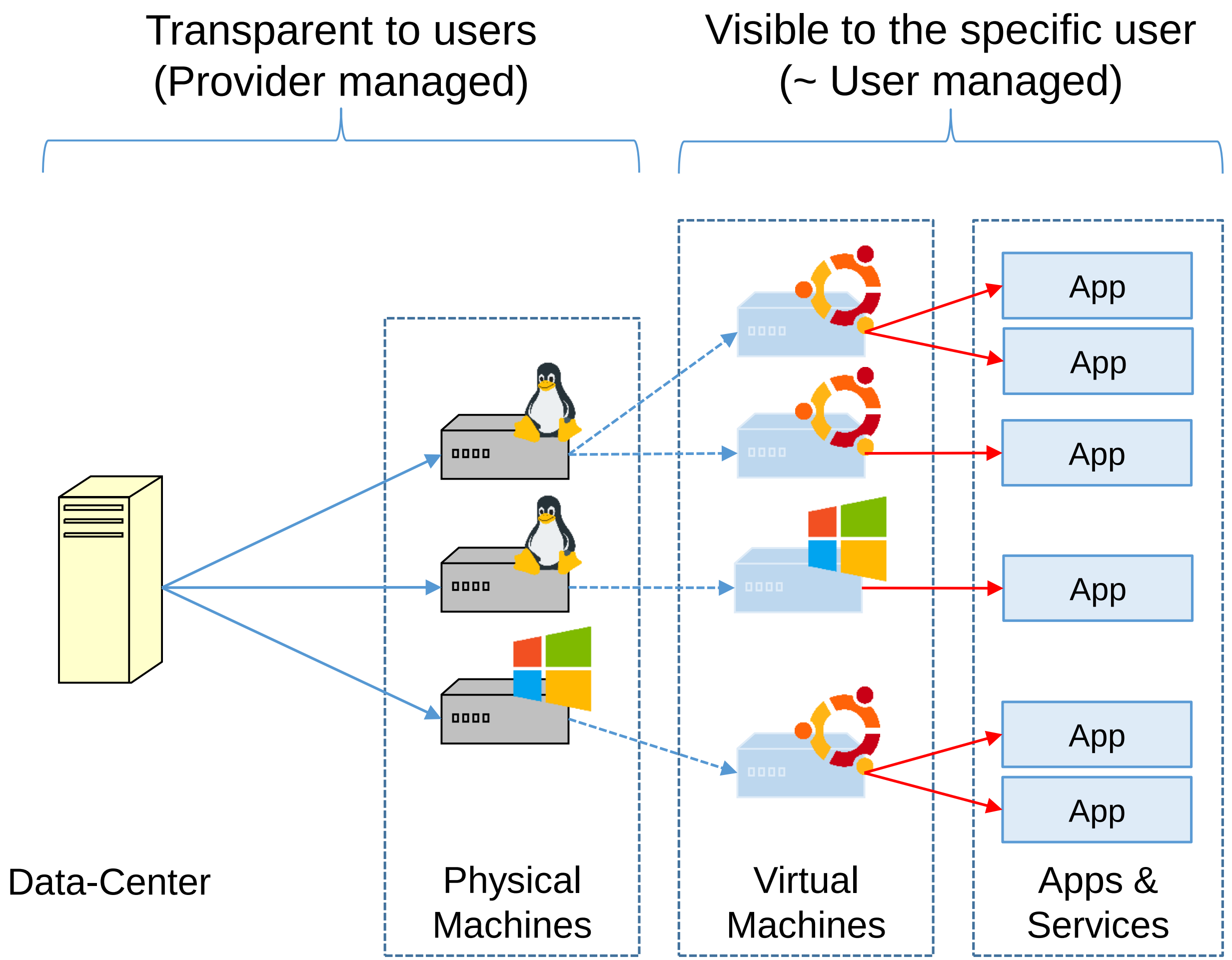


Sharing the HW

- Dimension your Environment
  - Description of the resources demanded by the VM
    - Resources are provided…
    - … but often not in exclusive → Resource Sharing
- Strategy of *Consolidation*
  - Packing VMs over resources
  - Sharing resources
    - Overbook resources
    - Leverage under-usage
    - Risk of performance decay
  - "Sharing resources"
    - Is the HV/OS/FW cleaning the footprints?
    - Beware: VM footprints might reveal our execution environment!
- Sharing Accelerators (GPUs and TPUs)
  - Some devices cannot be "virtualized"
    - GPUs and TPUs cannot be partitioned (yet)
    - A VM takes on a GPU and locks it until shutdown
  - Collision / Competition
    - Two applications attempt to use a GPU
      - They don't see the used memory of the other
      - They collision when uploading data
        - » "Not Enough Memory" Error
        - » "Data eviction" → Data is (unwantedly) removed to make room
    - Two VMs want to take on a GPU
      - (first) VM1 locks it at virtual boot
      - (second) VM2 cannot boot with the GPU

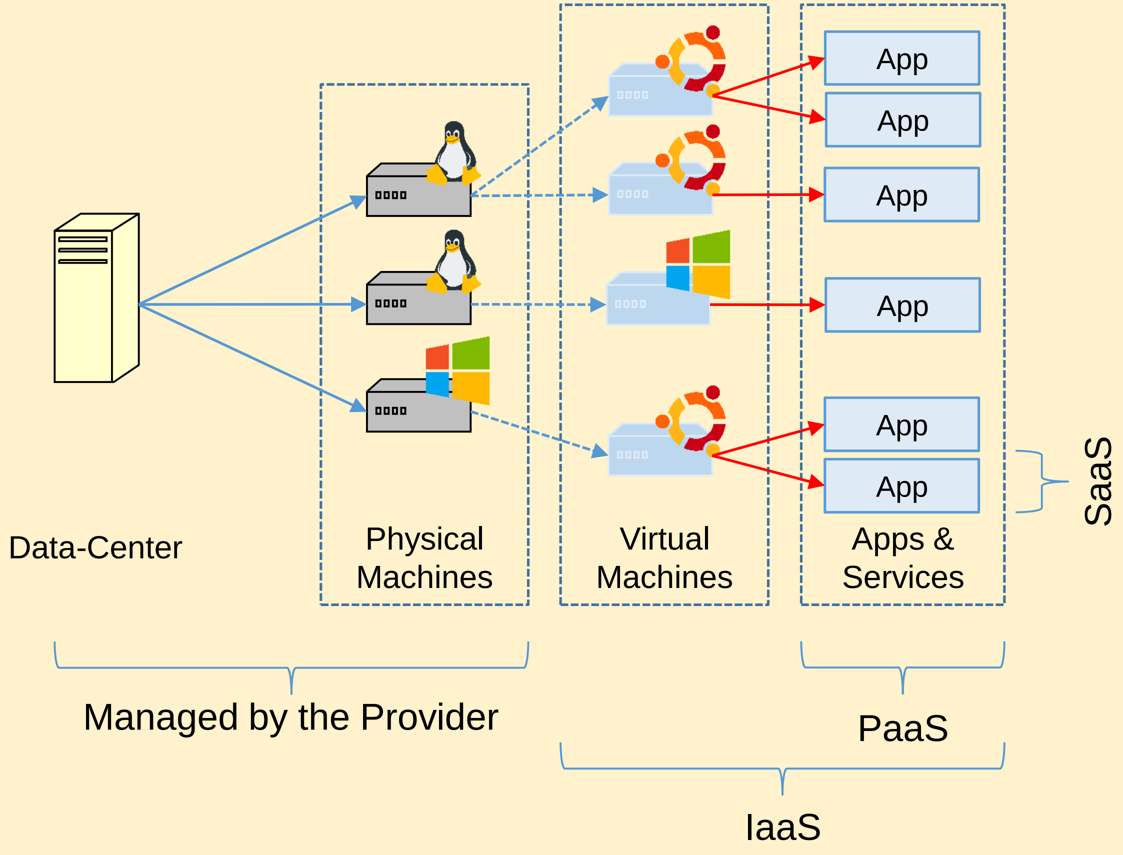
Virtualization in Resource Providers

- E.g., Clusters, Clouds and Data-Centers
  - Physical resources NOT provided
  - Only virtualized infrastructure



Virtualization in Resource Providers

- Clusters, Clouds and Data-Centers
  - Resource providers DO NOT give physical resources, but VMs!
    - IaaS / PaaS / SaaS
    - Software-Defined Infrastructure
    - No access to Host OS
    - Only access to Guest OSs / Platforms



Public Pre-Built VM Images

IMAGE REPOSITORIES

Image Repositories

- Prepared Images
  - Image Servers
  - Catalog → OS / SW / Services
    - Image → Template to copy & deploy
    - Instance → Deployed copy

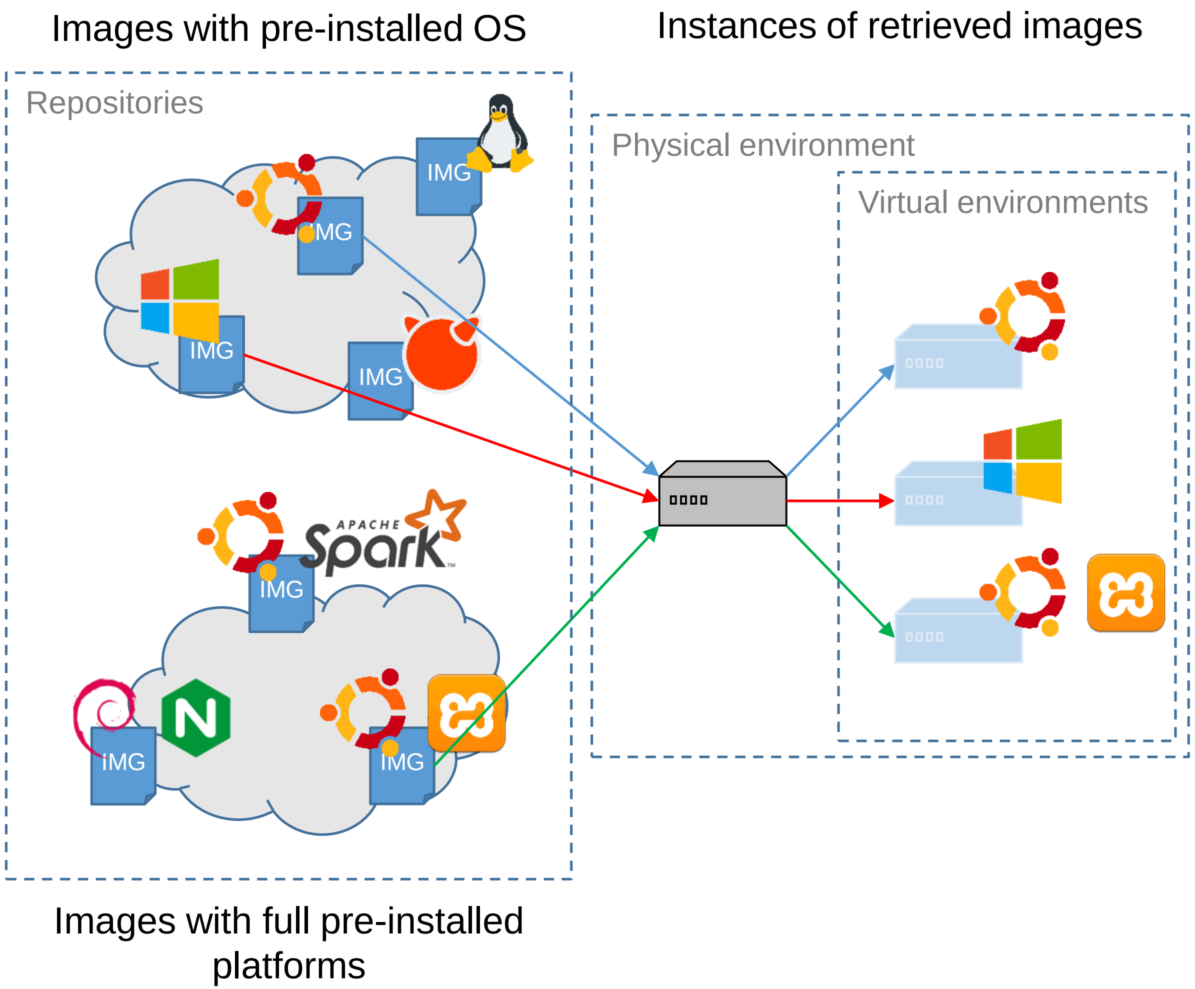
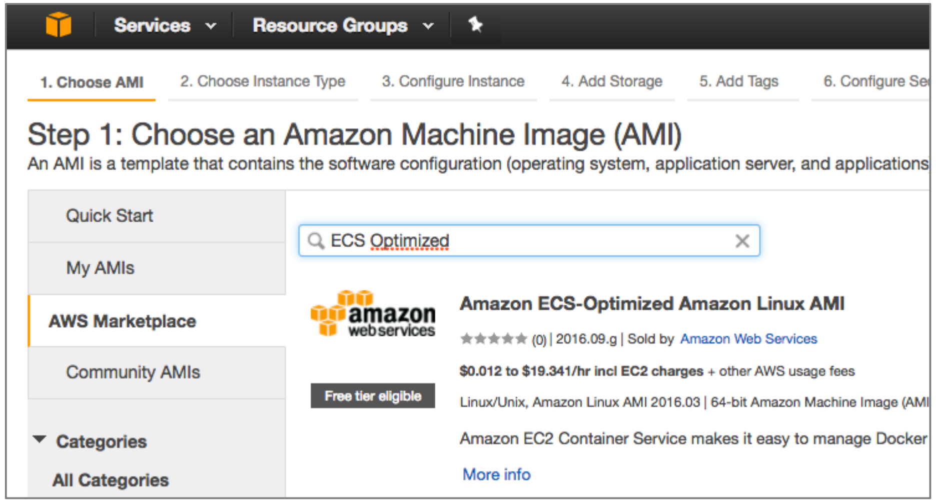


Image Repositories

- Use of Images
  - Cloud-provided
    - AWS & others: pre-created ready images
    - OpenStack: uploaded ready images
  - Custom images
    - Download & modify base images



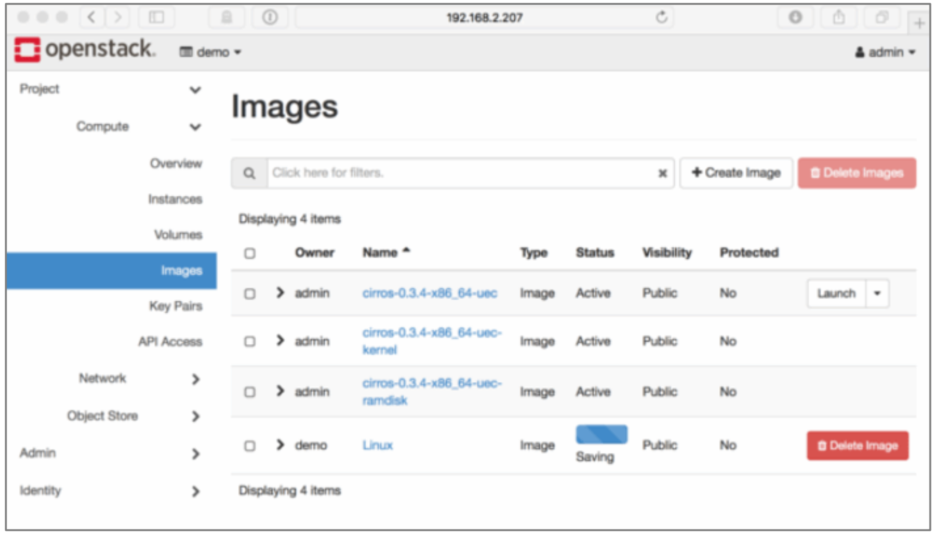


Image Repositories

- Prepared Images
  - Image Servers and Managers
    - Provide a catalog of pre-installed VMs with different SW Stacks and OS
    - Through image managers you can download an image, and deploy instances of it
  - Catalog
    - By Operating System (e.g., different versions of Ubuntu, Debian, …)
    - By SW stack (e.g. with Spark, Apache + PHP + MySQL, … pre-installed)
    - By deployed Services (e.g. with Python + Flask, Jupyter + Conda, … ready)

- Difference
  - Image → Template to copy & deploy (what you download)
  - Instance → Deployed copy (you create a copy of the image, and run it)

- Examples
  - Vagrant repositories
    - Different versions of Linux (Ubuntu, Debian, OpenSUSE, …)
    - Distributions with already installed software (XAMMP, Spark, Python+Pytorch, CUDA, …)
  - Custom repositories
    - Set up local repos at "home" → Modify base images and save them

VMs and Security

- Virtual Hijacking
  - Untrusted VM images/repositories → Malware!

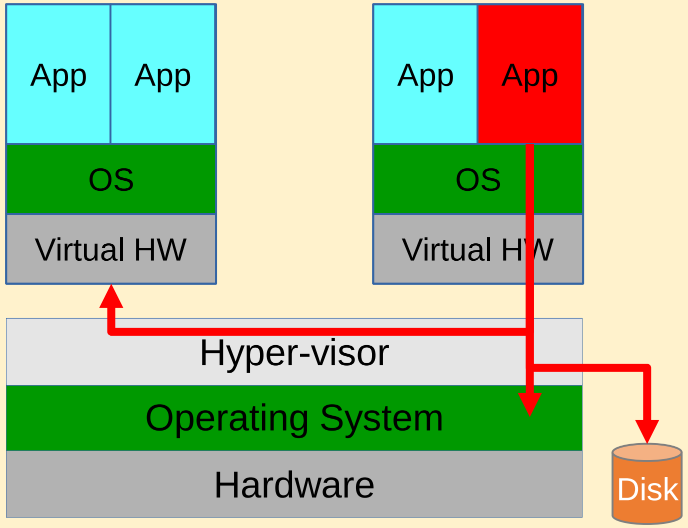
- Fundamental security
  - Constant update Host/Guest OS and Hypervisor
  - Use only trusted applications in VM
  - Control access to machines (PM & VM) and devices
  - Firewall/Control VM exposed services

THIS!

Threats to the VM/Hypervisor

- Virtual Hijacking
  - Unauthorized access to Hypervisor and VMM
    - Untrusted VM images/repositories
    - Zero-day exploits in Hypervisor and Host OS

- Access to VM connected devices
- Exploit Extended Page-Tables
- Etc...



- Baseline measures
  - Fundamental security
    - Keep updates Host/Guest OS and Hypervisor
    - Use only trusted applications in VM
    - Control user and application accesses to machines (both PMs & VMs) and devices
  - Networking and firewalling of the Virtual Network
    - Firewall and control VM exposed services (ports and IPs)

Threats to the VM/Hypervisor

- Unchecked environments
  - E.g., untrusted disks, images and software
    - Virtual MITM
    - Exploit execution, arbitrary code … against Hypervisor
  - E.g., virtualization nesting
    - Privilege acquisition in inner VMs
  - Potential solutions
    - Secure boot & TPM enabled
    - Trusted Virtual Disks/Images only
    - Avoid VM nesting

"Fake" VM hijacking a legitimate image

Inject code into the HV

"sudo" access to devices/files/services

Safe procedures for VM acquisition
Restricted permissions and VM monitoring

- Creative attacks
  - VM monitoring → Identification of VM content by resource consumption (memory access) and connections
  - VM snooping → Different tenant VMs in same network: eavesdropping content
  - Poisoning Memory → Overflow Hypervisor memory space: corrupt memory for VMs and host

Session Objectives

- Explain when/why we need to execute AI in scalable systems
- Explain how Virtual Machines work & how to use them
- Explain how image repositories help on exporting our AI software

Laboratori 7 – VMs

PRÀCTICA DE MÀQUINES VIRTUALS

Laboratori

- Entorn:
  - Servidor compartit CROMAI-FOG0{0,1}
  - VirtualBox → Hypervisor (gestiona i executa les VMs)
  - Vagrant → Repositori públic de VMs ja configurades

- Sistema Operatiu:
  - Debian 12 (Màquina hoste) / Ubuntu 22.04 (Màquina virtual) → Necessitem eines de sistema per monitoritzar la VM durant la pràctica

- Qüestionari:
  - Durant la pràctica cal resoldre preguntes respecte el que estem executant i observant

Laboratori

- Conèixer l'entorn de Virtualizació
  - Entrar en un sistema amb VBox i Vagrant (hypervisor i gestor d'imatges)
  - Entendre que els recursos estan compartits!
- Desplegar una VM
  - Descarregar una VM pre-configurada
  - Posar-la en execució i veure la diferència entre "dintre" i "fora"
  - Veure com consumeix recursos des de fora, com un procés
- Exposar serveis de dins la VM
  - Instal·lar un servidor web/aplicacions dins la VM
  - Veure com i des de on podem accedir al servei de la VM
- Desplegar una Xarxa Virtual
  - Engegar una segona VM
  - Configurar una xarxa virtual per comunicar les VMs
  - Veure com podem accedir als serveis de les VMs des de una altra VM i des de fora

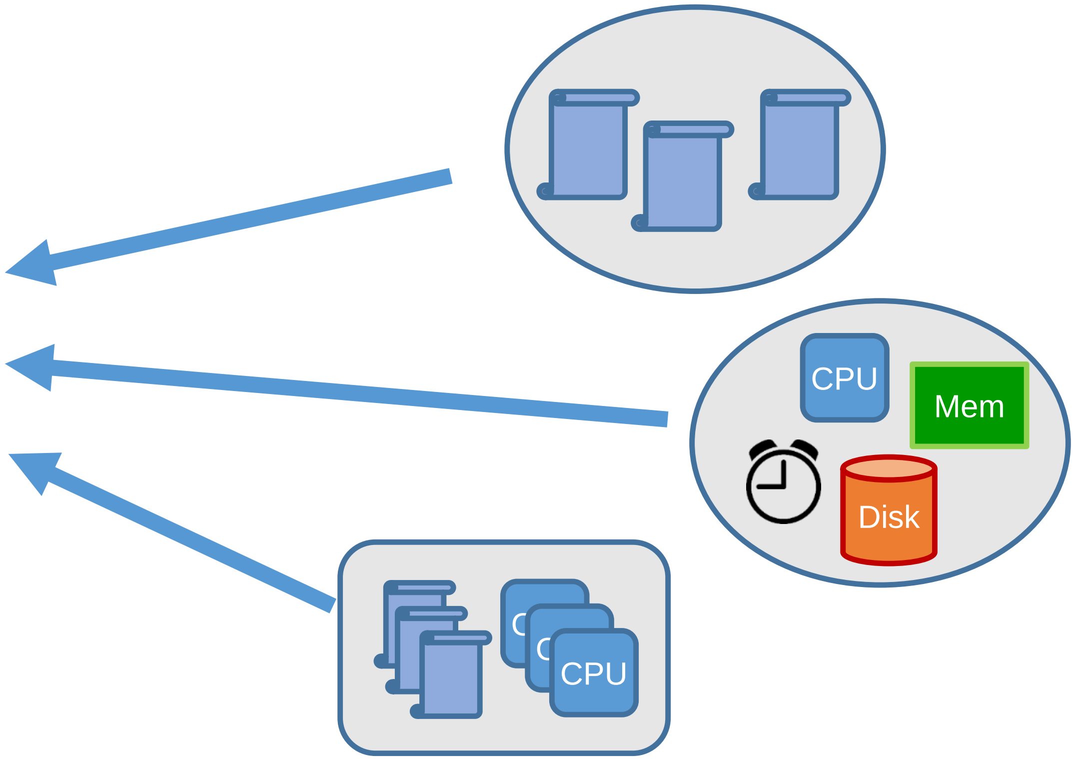
Auxiliary Slides

Little's Law

LOAD ESTIMATION

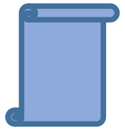
Little's Law

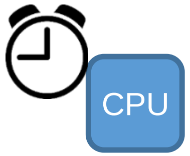
- Relation between
  - Received load
  - Resources/Time required
  - Average load // Required resources

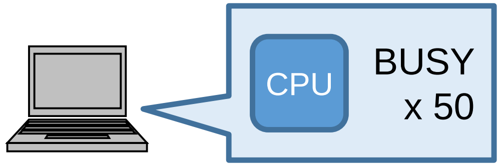


Little's Law: L = λW

- Example
  - We submit λ = 100 experiments / hour
  - Exps. take an average of W = 0.5 hours
    - [with 1 CPU per exp.]
  - Average number of exps. on our system: L = 50 exps
    - [avg. 50 CPUs in use]



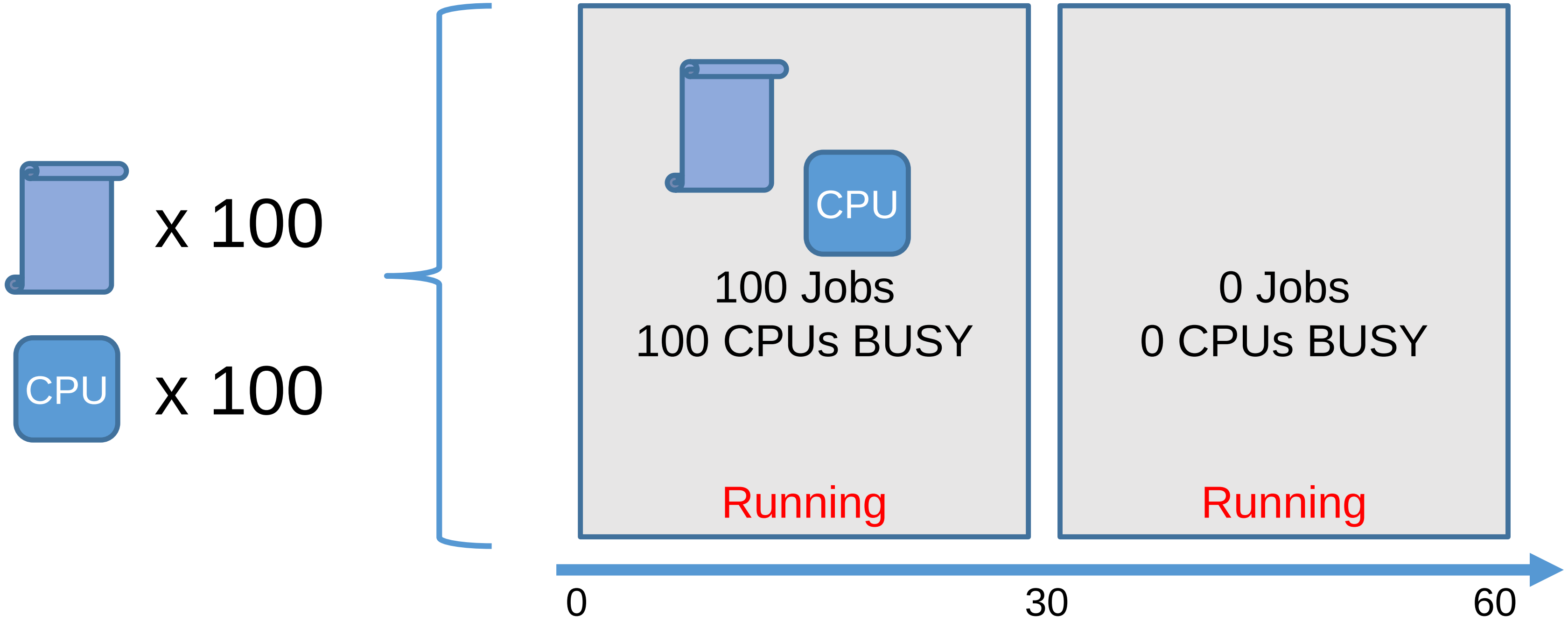




Resource Limits

- 100 CPUs

Arrival rate:	λ = 100 experiments / hour
Exps. take:	W = 0.5 hours [1 CPU per exp.]
Average exps. in:	L = 50 exps

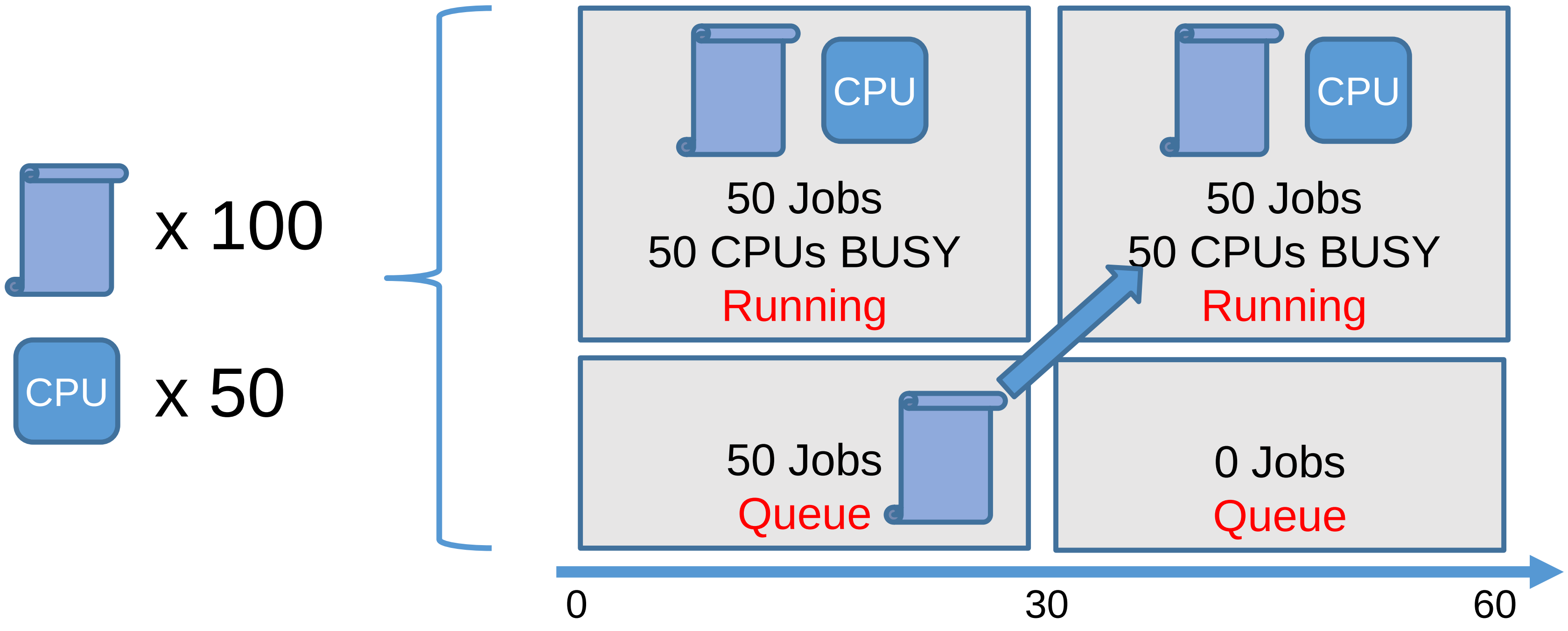


Resource Limits

- 50 CPUs

  (What Little's Law indicated)

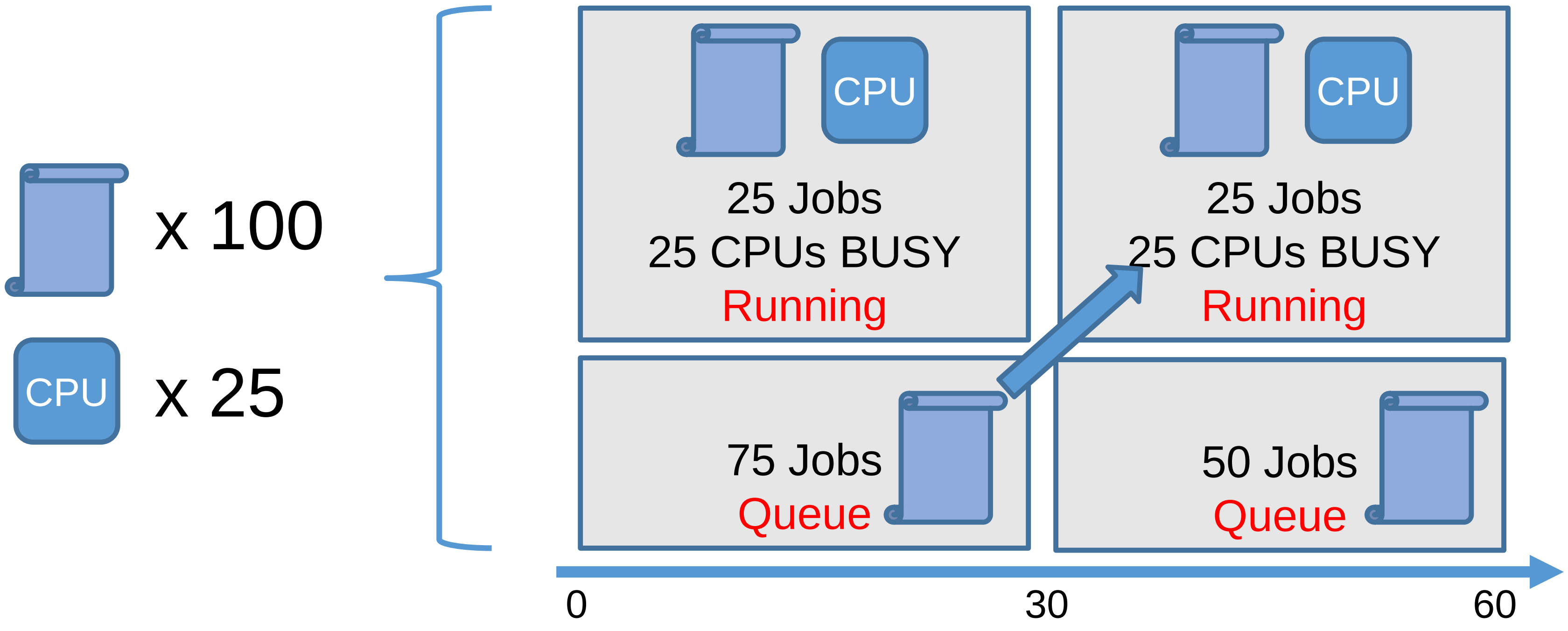
Arrival rate:	λ = 100 experiments / hour
Exps. take:	W = 0.5 hours [1 CPU per exp.]
Average exps. in:	L = 50 exps



Resource Limits

- 25 CPUs
  - Less than needed!

Arrival rate:	λ = 100 experiments / hour
Exps. take:	W = 0.5 hours [1 CPU per exp.]
Average exps. in:	L = 50 exps



Resource Limits

- 25 CPUs
  - Less than needed!

Arrival rate:	λ = 100 experiments / hour
Exps. take:	W = 0.5 hours [1 CPU per exp.]
Average exps. in:	L = 50 exps