Lecture 9

Cloud computing/ IoT and virtualization Techniques for Embedded Systems

Alessandro Vittorio Papadopoulos, Moris Behnam, Mohammad Ashjaei

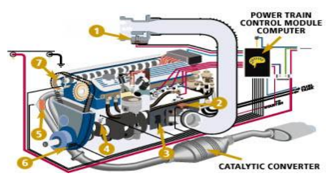
Mälardalen University

Outline

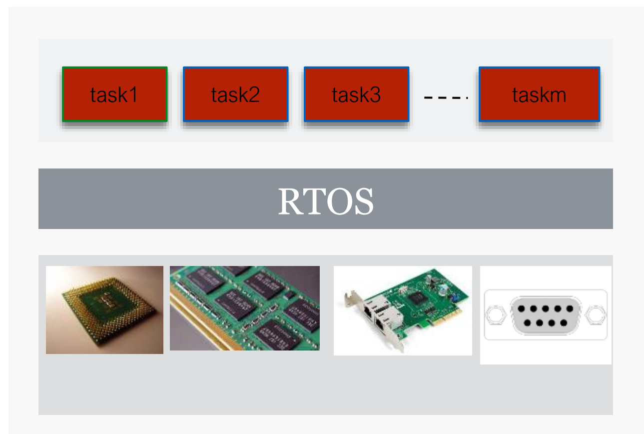
- **Introduction**
- **Part1**
  - **Cloud Computing**
  - **Fog and Edge Computing**
- **Part2**
  - **Virtualization**
  - **Hierarchical scheduling, CPU, Network**

Embedded Systems

- Software and hardware



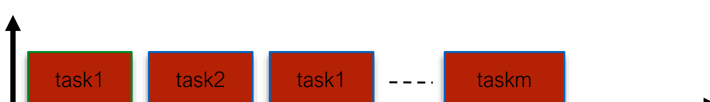
$task_1$ $task_2$ $task_3$ - - - $task_m$



Software

Operating system

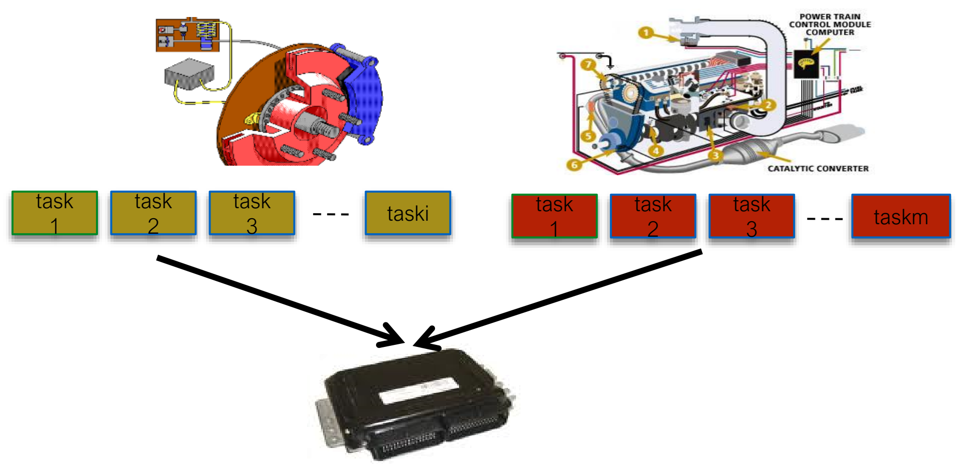
Physical resources



Scheduling

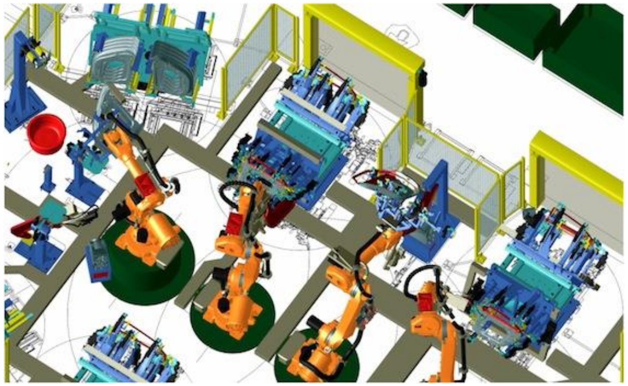
Cloud computing and Embedded Systems

- Increase of the size and functionalities of the embedded real-time distributed systems.
- The performance and the flexibility of the computing systems are increasing
- Software applications share the system resources including software and hardware resources

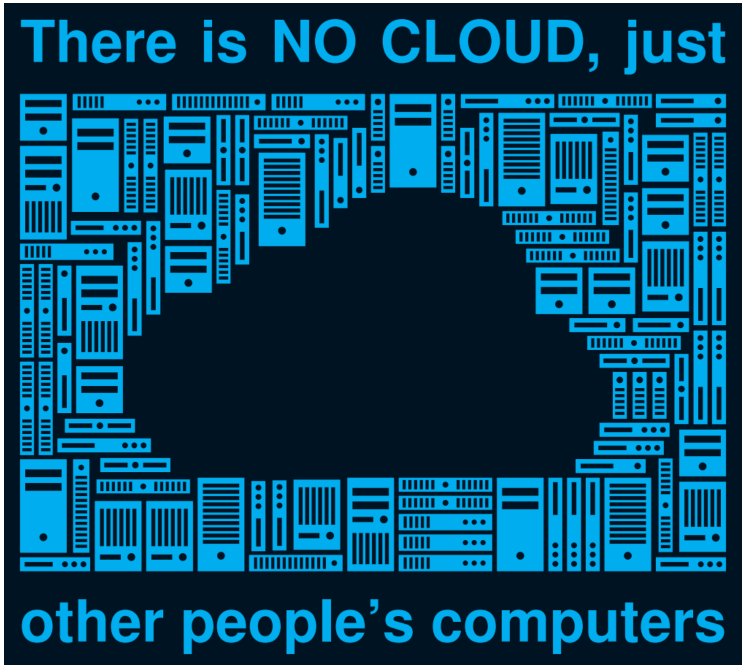


Cloud computing and Embedded Systems

- Sharing resources among software applications
- Example: Industrial Cloud Computing



Cloud computing



Introduction

- What is cloud computing?

https://www.youtube.com/watch?v=QJncFirhjPg

What is Cloud Computing?

National Institute of Standards and Technology (NIST)[1]:

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

- Essential characteristics:
  - On-demand self-service
  - Broad network access
  - Resource pooling
  - Rapid elasticity
  - Measured service

[1] https://www.nist.gov/programs-projects/cloud-computing

Service models

1. Cloud Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user- specific application configuration settings.

2. Cloud Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

3. Cloud Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

Service models (*aaS)

- SaaS: use services
- PaaS: develop/deploy services
- IaaS: host services

Cloud ‘stack’	**Target customer**
SaaS	End user
PaaS	Developers
IaaS	Operators/IT

Deployment models

- **Private cloud.** The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

- **Community cloud.** The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

- **Public cloud.** The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

- **Hybrid cloud.** The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

What's inside?

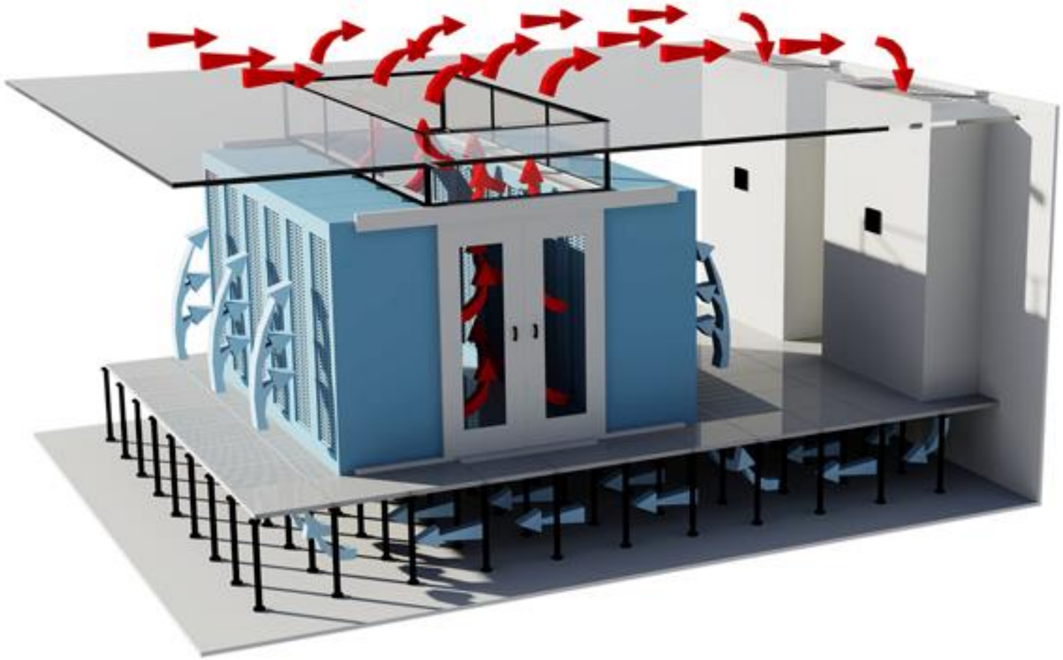
Racks

What's inside?

- Energy costs for data centres already exceed $15 billion/year.
- 10.4% annual growth rate
- In 2014
  - 70 billion kWh
  - 2% of total US electricity consumption

Power supplies

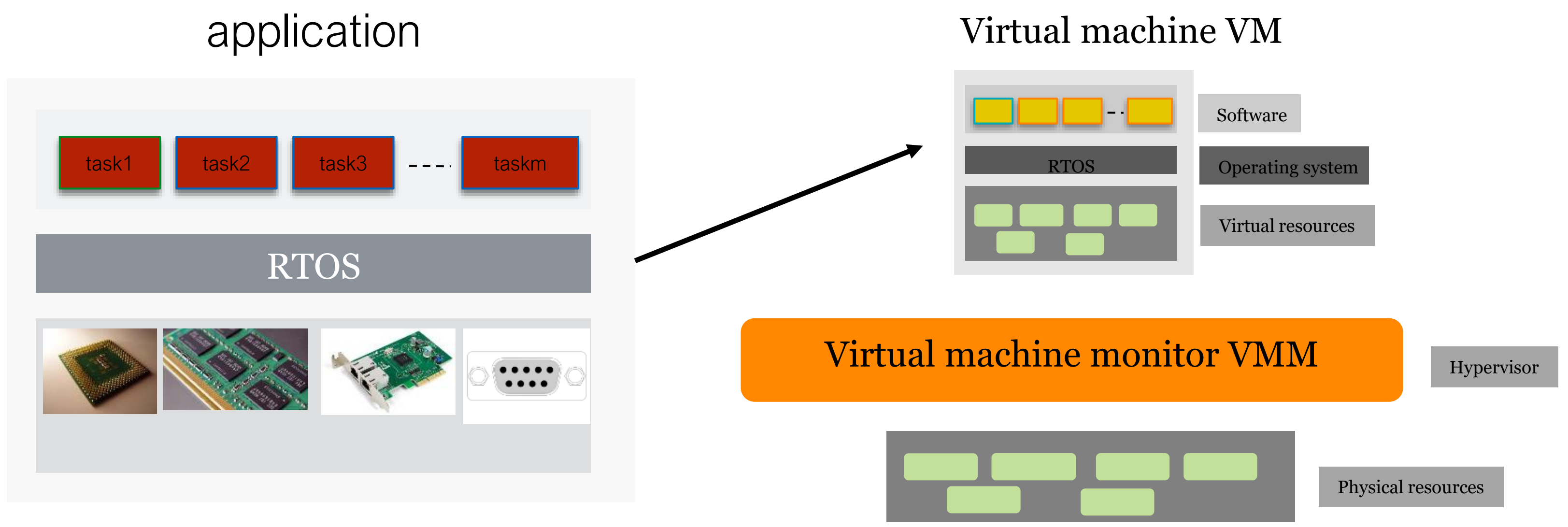
What's inside?



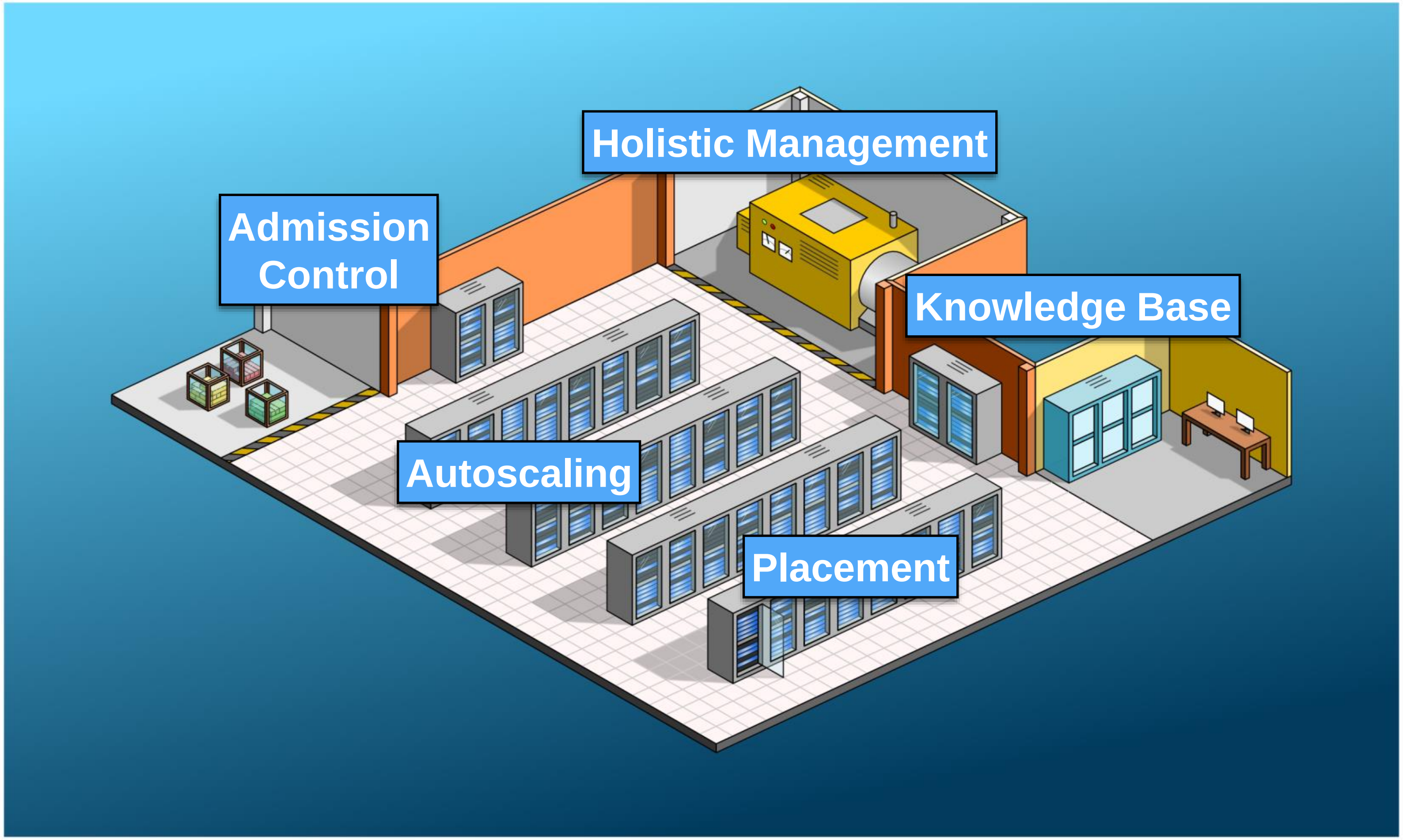
Cooling

Virtual machines

- Applications executed in virtual machines

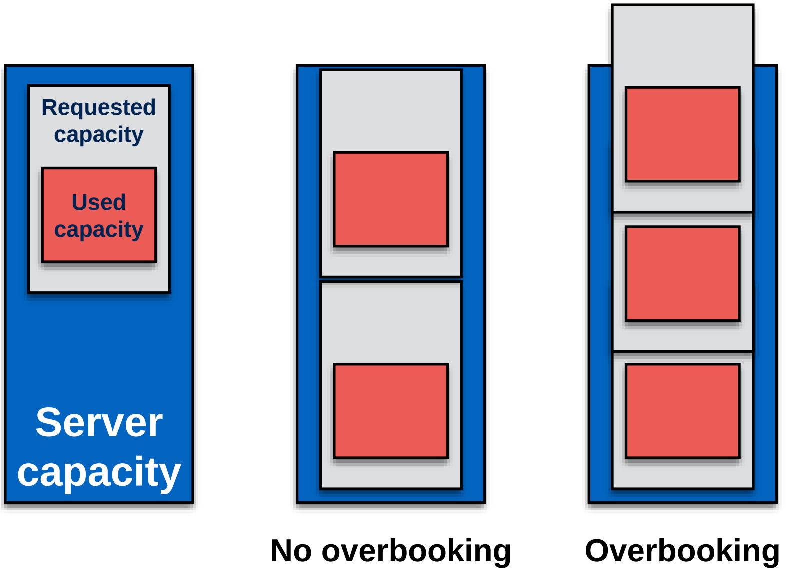


Problem classification



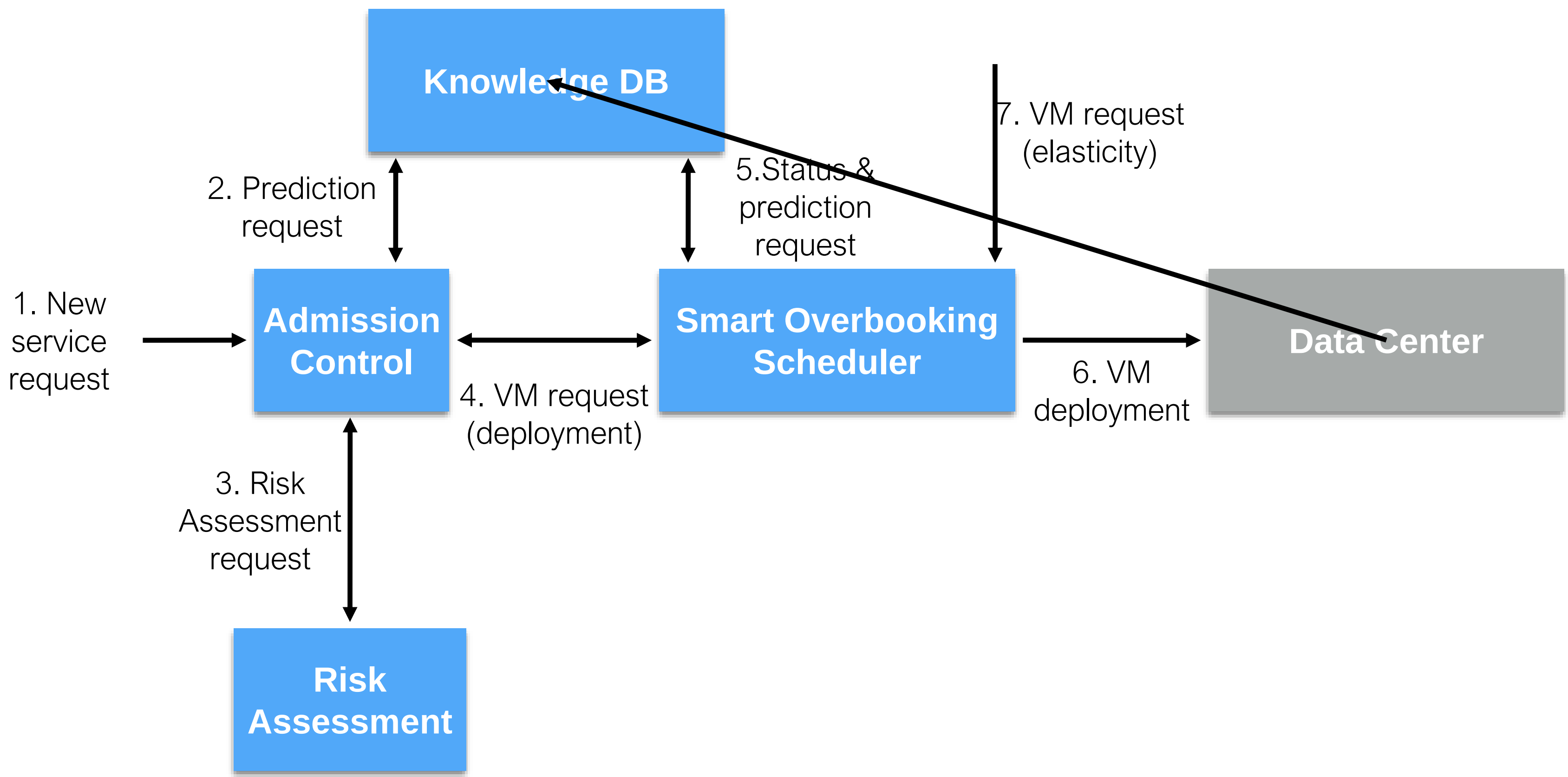
Admission Control

- Determine load, revenue, and risk
- Risk theory
  - Utility vs Violations
  - Overbooking
  - Long term effects
- **Overbooking**[1]

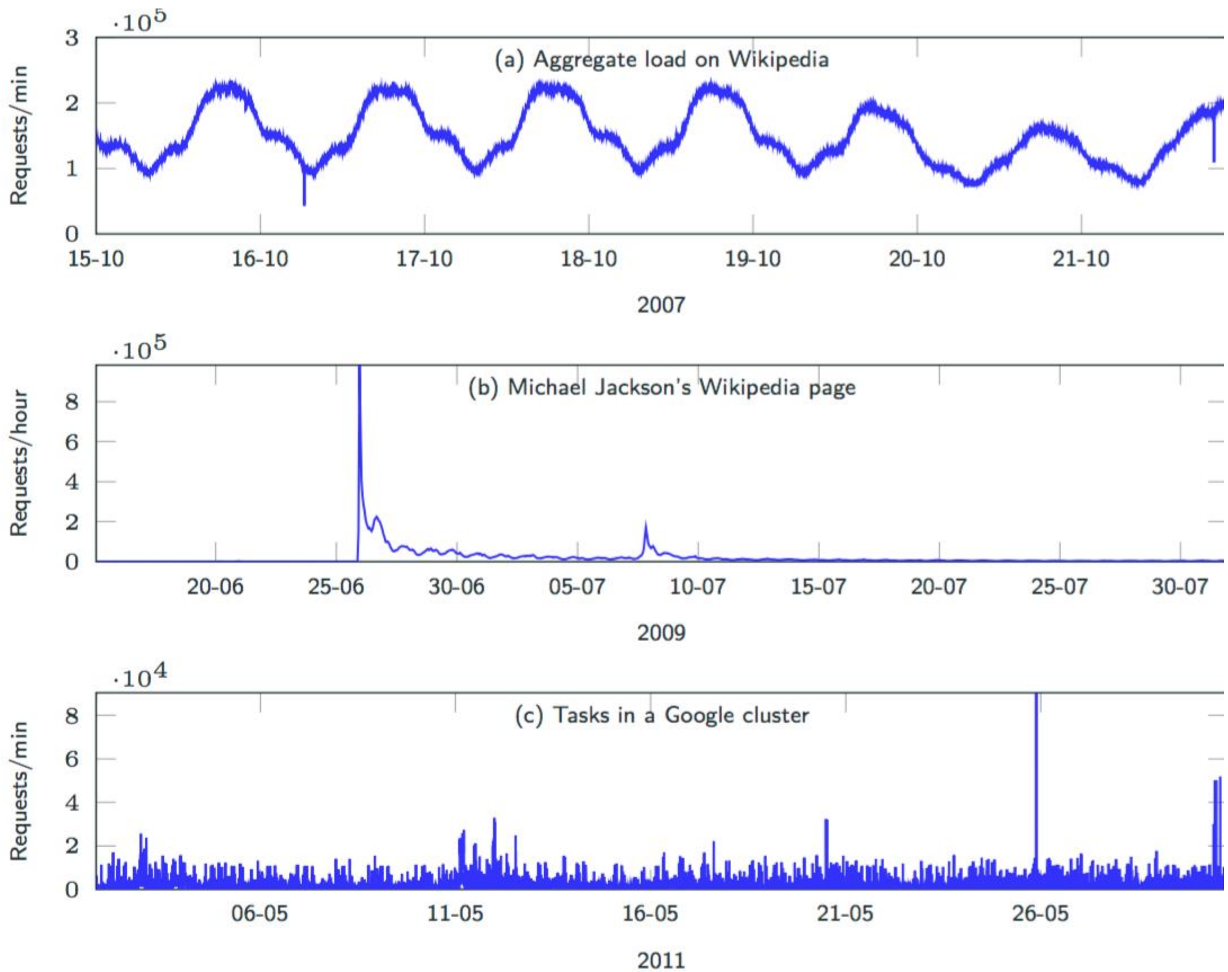


1 Tomás and Tordsson (2013). "Improving cloud infrastructure utilization through overbooking".

Overbooking



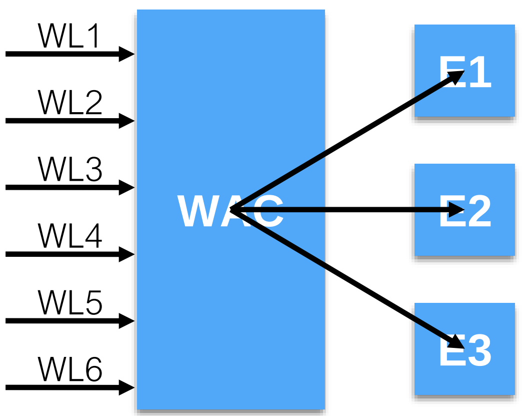
Workloads



Papadopoulos et al. (2016), “PEAS: A Performance Evaluation Framework for Auto-Scaling Strategies in Cloud Applications”.

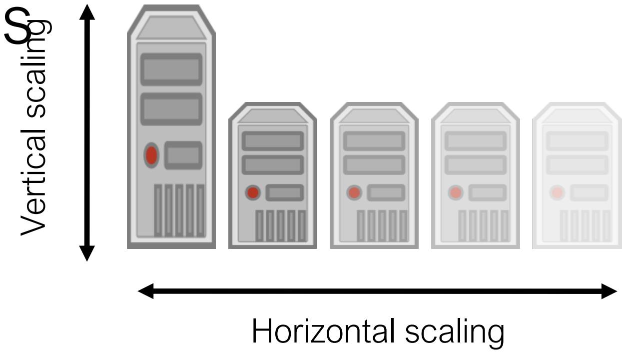
Capacity autoscaling

- **Aspects of the problem**
  - Meet variations in request rate
  - Meet variations in system response
  - Irregular vs planned workload
  - Multiple time-scale
  - KPIs
  - Signal vs. noise
  - Spikes
- **Auto-selection of workloads**
  - Workload classification
  - Application classification



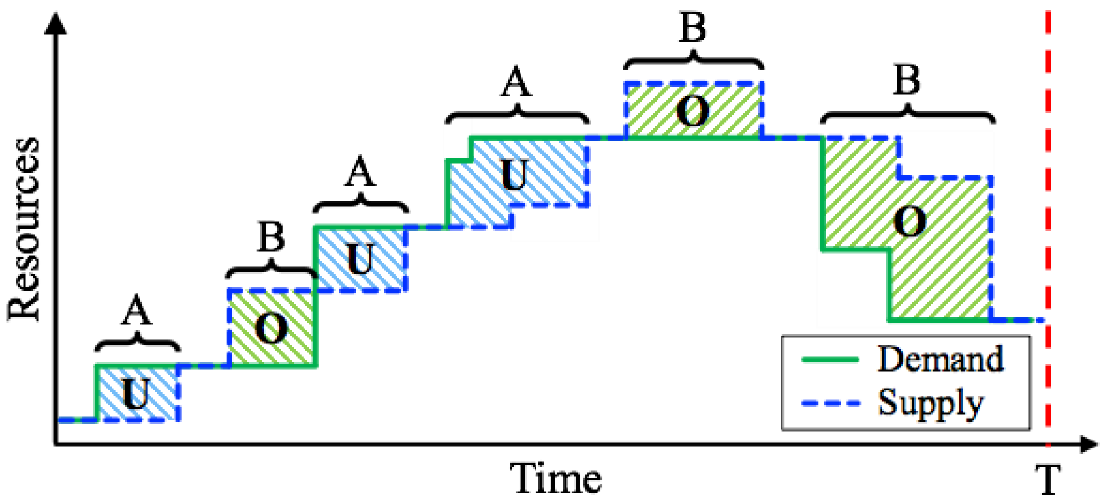
Autoscaling (Elasticity)[1]

- **Workload analysis**
  - Typology of applications
  - Spike detection
- **Scaling methods**
  - Horizontal/Vertical
  - Reactive/Predictive



[1] Herbst et al. (2013) "Elasticity in Cloud Computing: What It Is, and What It Is Not".
url: https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst

Under- and Over-Utilisation



Accuracy of under- and over-utilization[1]

$$a_U := \frac{1}{T \cdot R} \sum_{t=1}^{T} (d_t - s_t)^+ \Delta t,$$

$$a_O := \frac{1}{T \cdot R} \sum_{t=1}^{T} (d_t - s_t)^+ \Delta t.$$

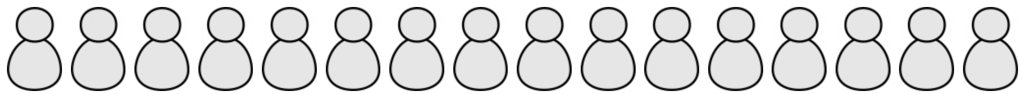
Many other metrics can be defined to evaluate the autoscaler performance

[1] Ilyushkin et al., “An Experimental Performance Evaluation of Autoscaling Algorithms for Complex Workflows”.

Open source (horizontal) autoscaling algorithms

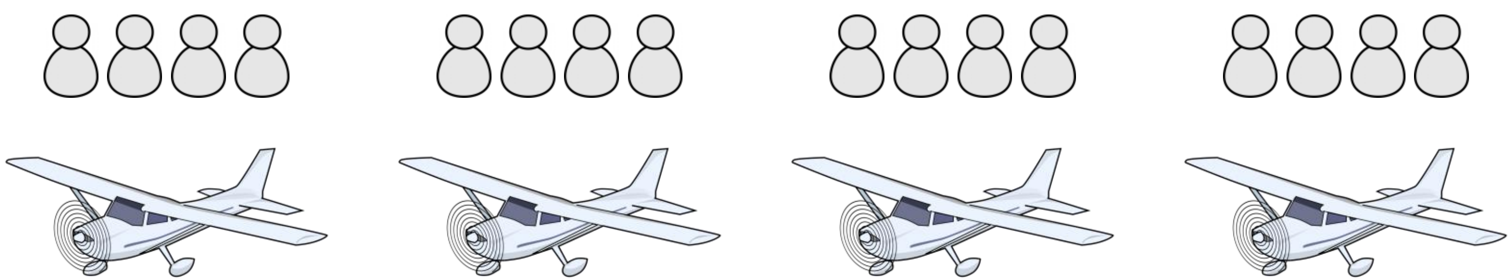
- **Hist**: Urgaonkar et al. (2008)
- **AutoScale**: Gandhi et al. (2012)
- **AGILE**: Nguyen et al. (2013)
- **ElastMan**: El-Shishtawy and Vlassov (2013)
- **Adapt**: Ali-Eldin et al. (2013)
- **ConPaaS**: Fernandex et al. (2012)
- …

Horizontal vs Vertical Elasticity

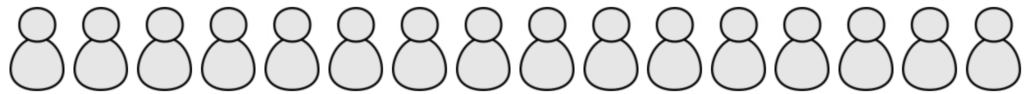
Application: 

Host:  **(too small)**

Scale Horizontally

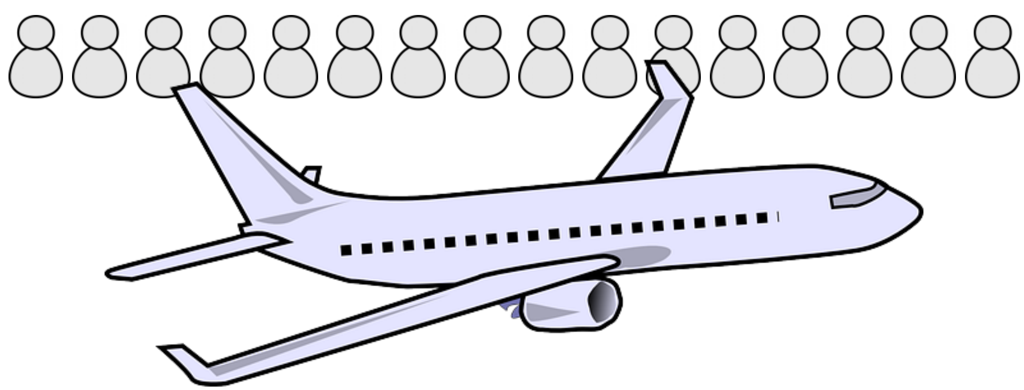


Horizontal vs Vertical Elasticity

Application: 

Host:  **(too small)**

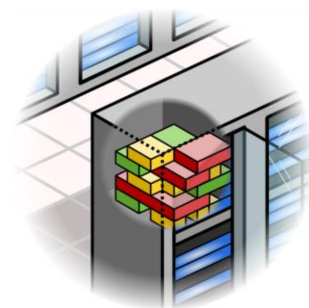
Scale Vertically



Lakew et al. (2017) “KPI-agnostic Control for Fine-Grained Vertical Elasticity”

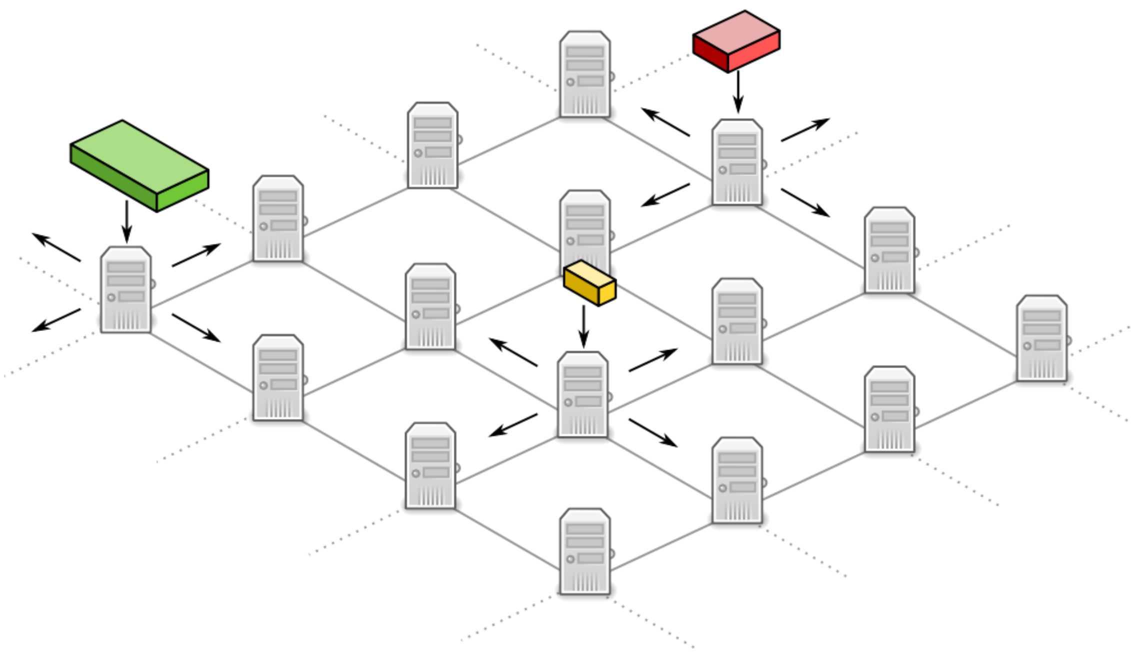
Placement

- VM Placement
- Application Placement
- VM Migration



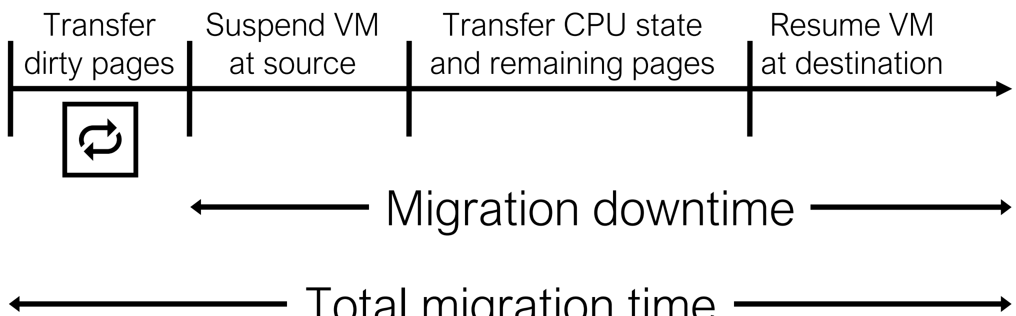
VM placement

- **Map VMs to resources**
  - After admission
  - After scaling
  - To reconsolidate
- **Inter-datacenter**
  - Optimisation
  - Heuristics
- **Intra-datacenters**
  - Multi-dimensional multi-knapsack problem

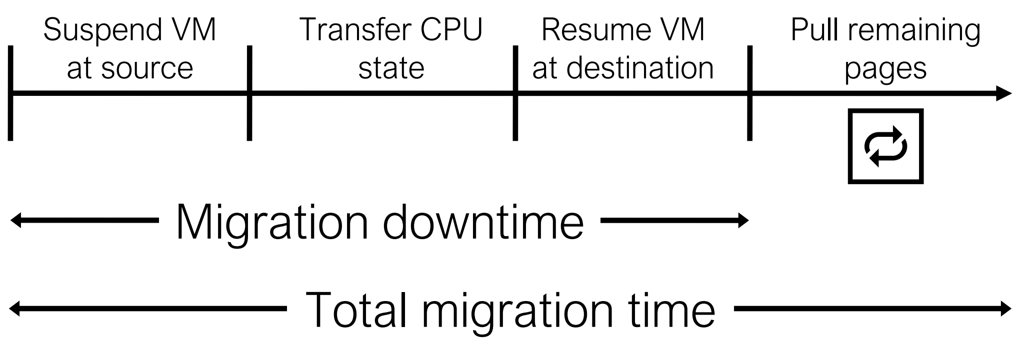


Live VM migration

Pre-copy migration

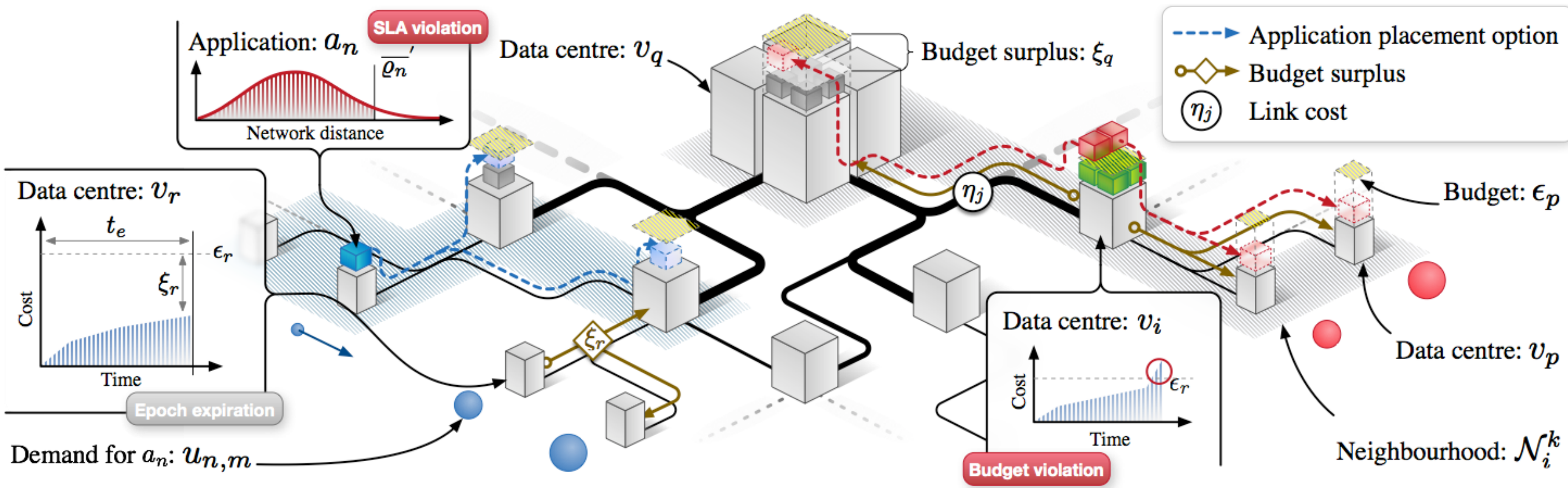


Post-copy migration



	Pre	Post
Continuous service		(✓)
Resource usage		✓
Robustness	✓	
Predictability		✓
Transparency	✓	✓

Application placement "in big"



Tärneberg et al., (2017) "Distributed Approach to the Holistic Resource Management of a Mobile Cloud Network"

Other relevant problems

- Knowledge base
  - Monitoring
  - Accounting
  - Energy

- Dynamic resource rationing
  - QoS-level adherence
  - Overall cost-benefit with QoS-level weights
  - Constrained optimisation
  - System feedback on KPI and dimmer effect
  - Graceful degradation and resiliency

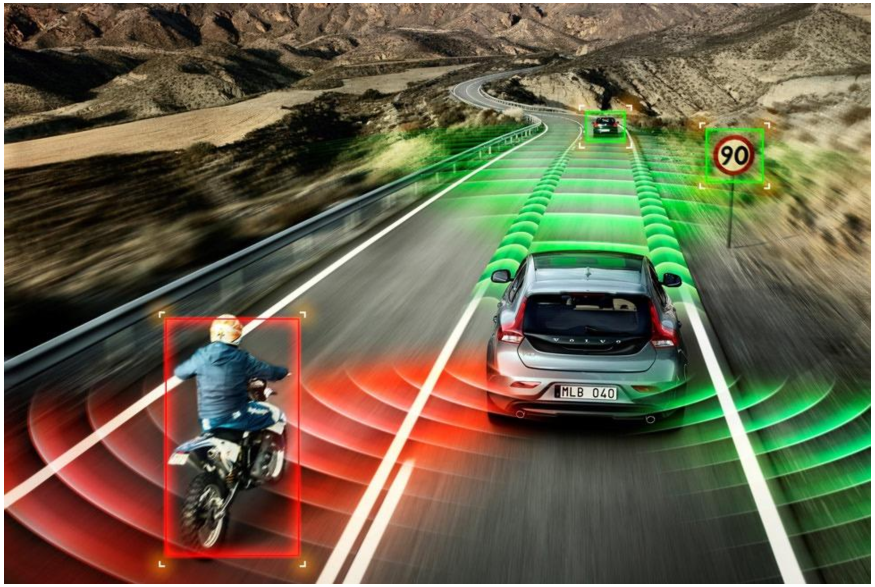
- Holistic management

Cloud Computing Limitations

- **Connectivity** is a prerequisite
- Cloud computing assumes that there is **enough bandwidth** to collect the data
- Cloud computing centralizes the analytics thus defining the lower bound **reaction time** of the system

Not always connected

- Some IoT systems **need to be able to work even when connection is temporarily unavailable** or under degraded conditions
- Driver assistance applications can’t rely on a connection to the cloud to be always available
- Cloud computing is useful in offloading some computations, but other decisions that require short reaction time, or that have to be taken in a decentralized fashion can’t rely on the cloud

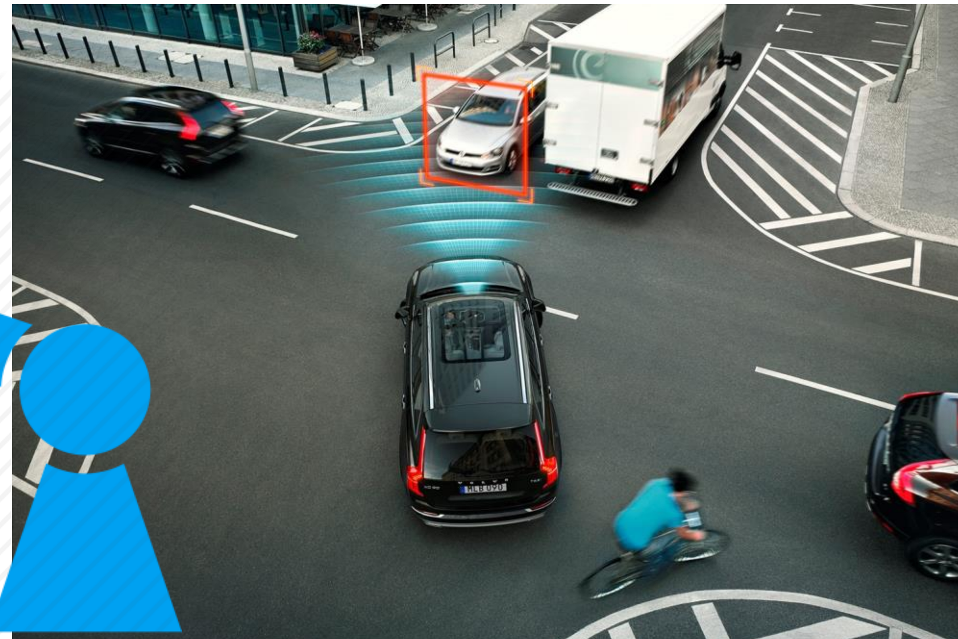


Not always enough bandwidth

- It can become an overly strong assumption for Industrial IoT (IIoT) applications
- For some IIoT applications it is just not realistic to stream all data to a cloud

Application	Required bandwidth
Energy Utility Co.	0.5TB/day
Offshore Oil Field	0.75TB/week
Large refinery	1TB/day
Airplane	10 TB/30 min of flight

Not always reactive



- Some IoT applications won't be able to **wait for the data to get to the cloud,** be analyzed and for insights to get back

Fog and Edge Computing

Fog Computing vs Edge Computing

"Both fog computing and edge computing involve pushing intelligence and processing capabilities down closer to where the data originates" from pumps, motors, sensors, relays, etc."

"The key difference between the two architectures is exactly where that intelligence and computing power is placed"

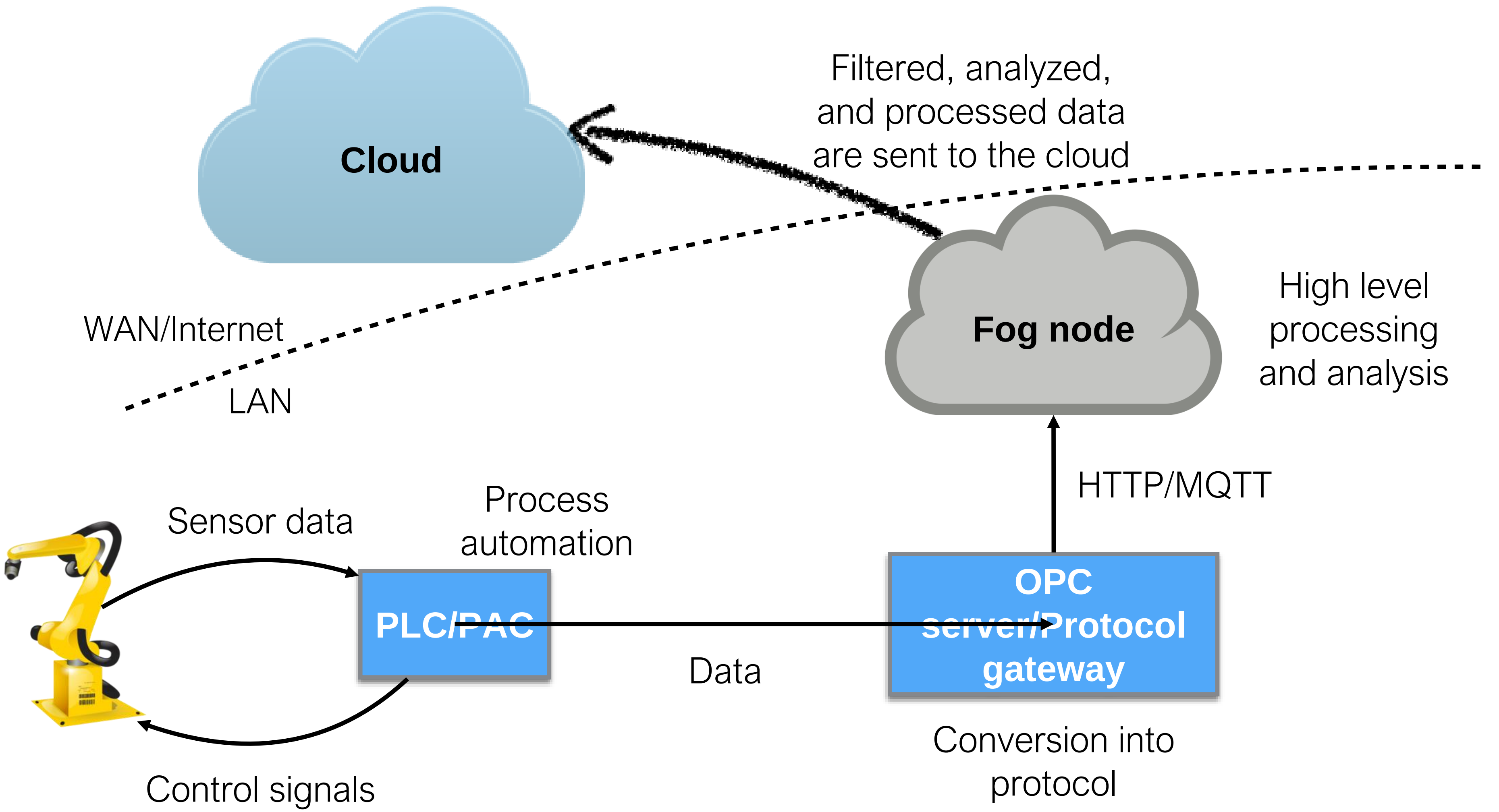
Matt Newton, director of technical marketing at Opto 22

Fog Computing vs Edge Computing

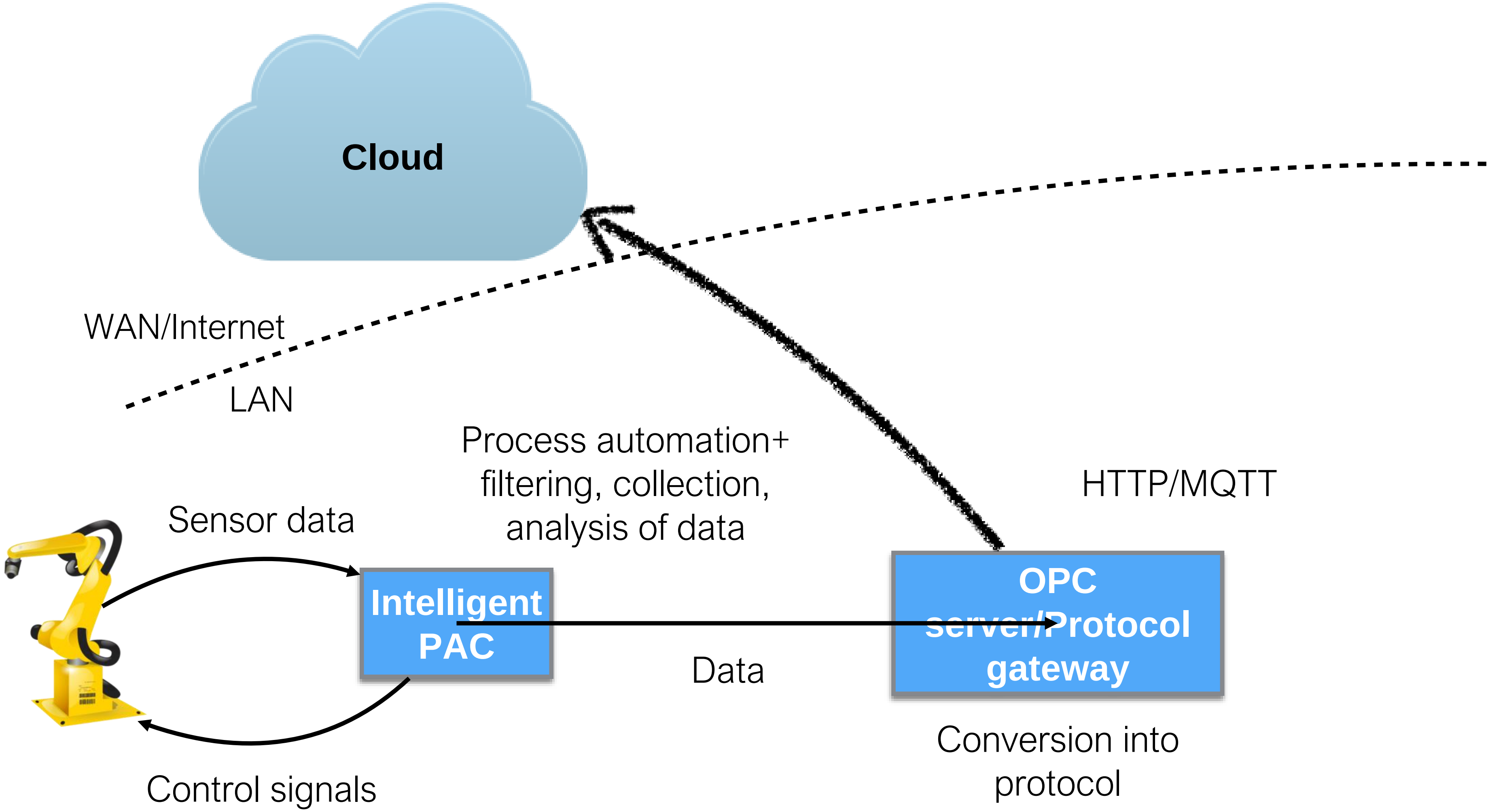
- **Fog computing** pushes intelligence down to the local area network (LAN) level of network architecture, processing data in a fog node or IoT gateway.

- **Edge computing** pushes the intelligence, processing power, and communication capabilities of an edge gateway directly into devices like PACs (programmable automation controllers).

Fog Computing



Edge Computing



What is “Fog Computing”

“The fog extends the cloud to be closer to the things that produce and act on IoT data. These devices, called fog nodes, can be deployed anywhere with a network connection: on a factory floor, on top of a power pole, alongside a railway track, in a vehicle, or on an oil rig. Any device with computing, storage, and network connectivity can be a fog node. Examples include industrial controllers, switches, routers, embedded servers, and video surveillance cameras.”

Cisco, “Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are”

OpenFog Consortium

- Consortium of high tech industry companies and academic institutions across the world aimed at the standardization and promotion of fog computing

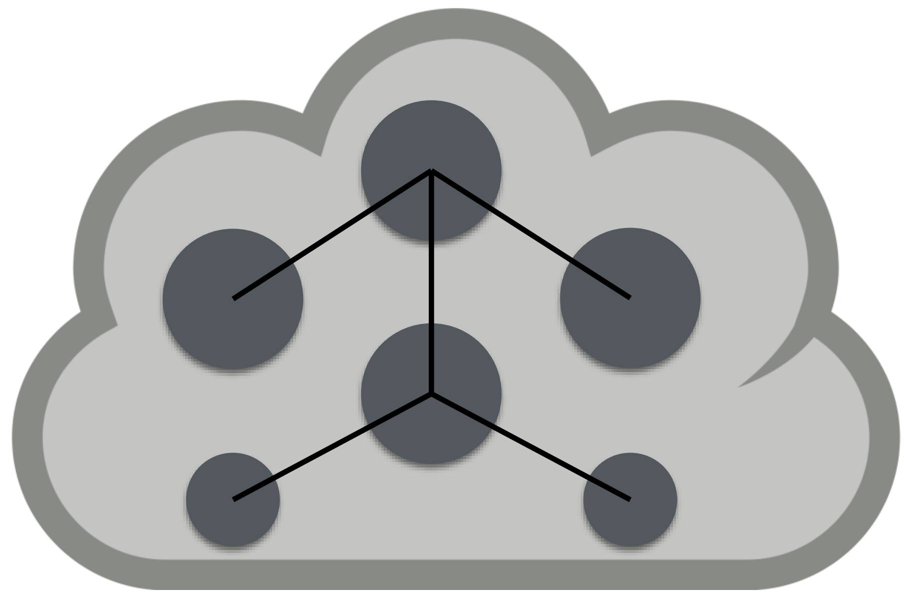
https://www.openfogconsortium.org/

Why fog?

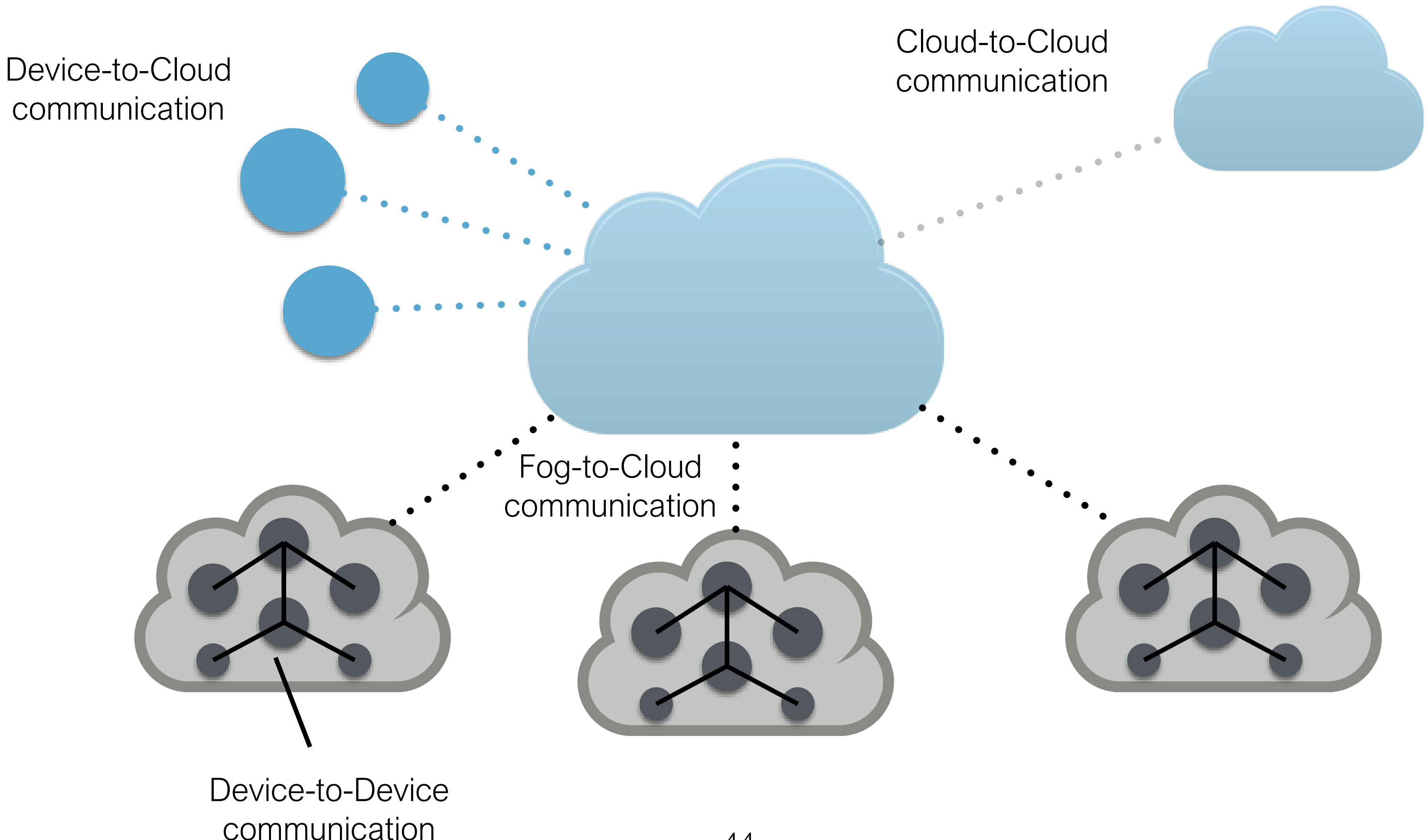
- Whereas the **cloud is “up there”** in the sky somewhere, distant and remote and deliberately abstracted, the **“fog” is close to the ground, right where things are getting done.**

What is Fog Computing

- Fog computing is about computing on the edge
- In Fog computing devices communicate peer-to-peer to efficiently share/store data and take local decisions



Synergy of Cloud and Fog computing



Goals of fog computing

1. To improve efficiency and trim the amount of data that requires to be transmitted for processing, analysis and storage.

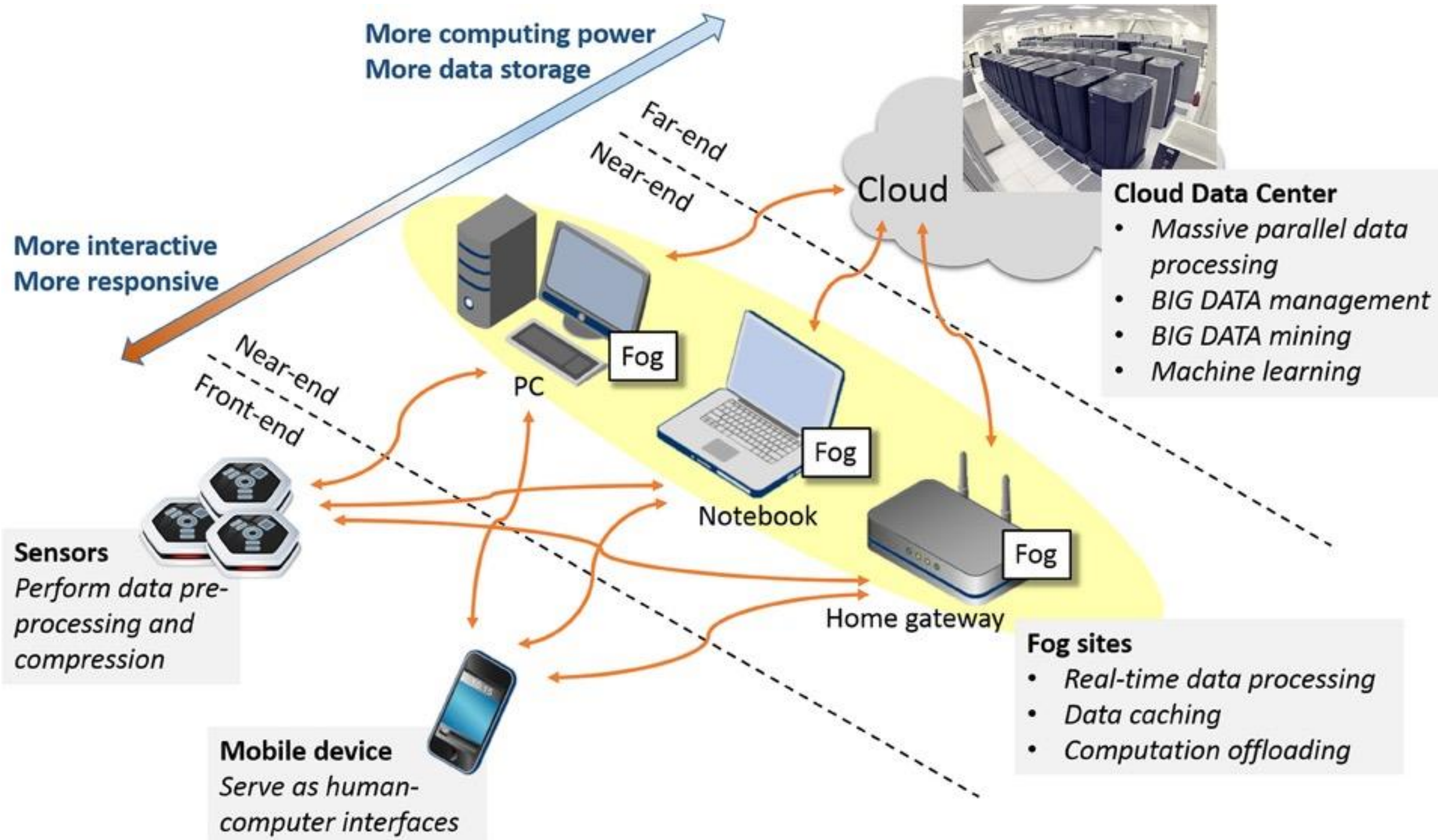
2. Place the data close to the end user.

3. Provide security and compliance to the data transmission over cloud.

Characteristics

- Edge location, location awareness, and low latency
- Geographical distribution
- Support for mobility
- Real-time interactions
- Heterogeneity
- Interoperability

Fog architecture



When to consider Fog Computing

- Data is collected at the extreme edge: vehicles, ships, factory floors, roadways, railways, etc.
- Thousands or millions of things across a large geographic area are generating data.
- It is necessary to analyze and act on the data in less than a second.

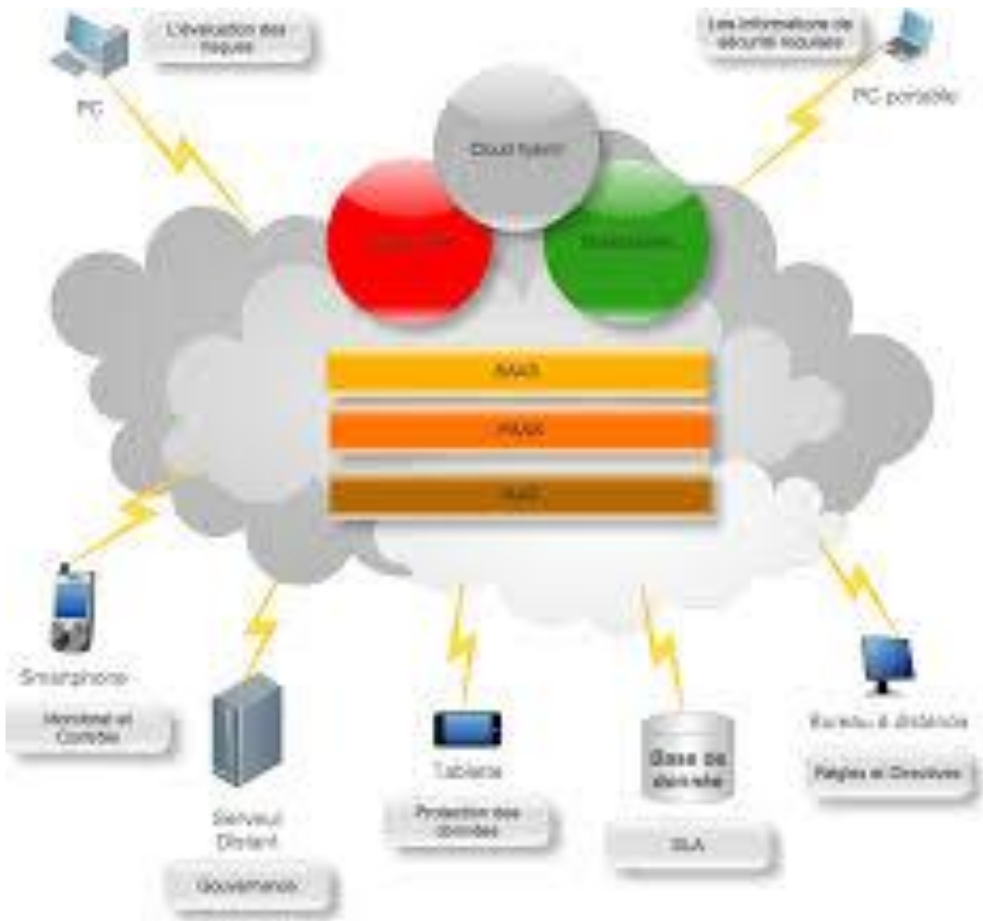
Fog vs Cloud

	Fog Nodes closest to IoT devices	Fog aggregation nodes	Cloud
Response time	Milliseconds to subsecond	Seconds to minutes	Minutes, days, weeks
Application examples	M2M communication Haptics, including telemedicine and training	Visualization Simple analytics	Big data analytics Graphical dashboards
How long IoT data is stored	Transient	Short duration: perhaps hours, days, or weeks	Months or years
Geographic coverage	Very local: for example, one city block	Wider	Global

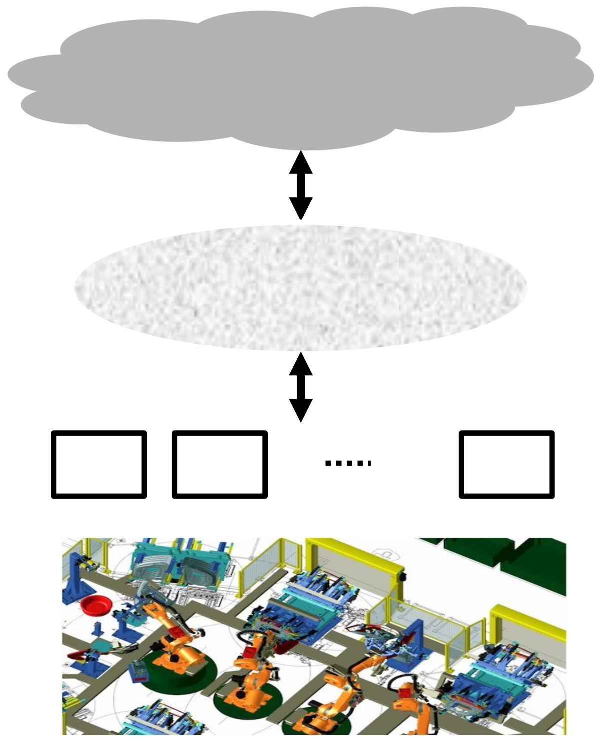
General References on Cloud

- Al-Shishtawy and Vlassov (2013). “ElastMan: Autonomic Elasticity Manager for Cloud-based Key-value Stores”. In: HPDC ’13. doi: 10.1145/2462902. 2462925
- Ali-Eldin et al. (2012). “An adaptive hybrid elasticity controller for cloud infrastructures”. In: NOMS, doi: 10.1109/NOMS.2012.6211900
- Caron et al. (2010). “Forecasting for Grid and Cloud Computing On-Demand Resources Based on Pattern Matching”. In: CloudCom. doi: 10.1109/CloudCom.2010.65
- Ali-Eldin et al. (2014). “How Will Your Workload Look Like in 6 Years? Analyzing Wikimedia’s Workload”. In: IC2E doi: 10.1109/IC2E.2014.50
- Gandhi et al. (2012). “AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers”. In: ACM ToCS. doi: 10.1145/2382553.2382556.
- Herbst et al. (2013). “Elasticity in Cloud Computing: What It Is, and What It Is Not”. In: ICAC url: https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst
- Ilyushkin et al. (2017). “An Experimental Performance Evaluation of Autoscaling Algorithms for Complex Workflows”. In: ICPE. url: http: //www.es.mdh.se/publications/4648-
- Lakew et al. (2017). “KPI-agnostic Control for Fine-Grained Vertical Elasticity”. In: CCGrid. url: http://www.es.mdh. se/publications/4649-
- Nguyen et al. (2013). “AGILE: Elastic Distributed Resource Scaling for Infrastructure-as-a-Service”. In: ICAC url: https://www.usenix.org/conference/icac13/technical-sessions/ presentation/nguyen
- Papadopoulos et al. (2016). “PEAS: A Performance Evaluation Framework for Auto-Scaling Strategies in Cloud Applications”. In: ACM TOMPECS. doi: 10.1145/2930659
- Pierre and Stratan (2012). “ConPaaS: A Platform for Hosting Elastic Cloud Applications”. In: IEEE Internet Computing doi: 10.1109/MIC.2012.105.
- Tärneberg et al. (2017). “Distributed Approach to the Holistic Resource Management of a Mobile Cloud Network”. In: ICFEC. url: http://www.es.mdh. se/publications/4683-
- Tomas and Tordsson (2013). “Improving Cloud Infrastructure Utilization Through Overbooking”. In: CAC. doi: 10.1145/2494621.2494627
- Urgaonkar, Bhuvan et al. (2008). “Agile Dynamic Provisioning of Multi-tier Internet Applications”. In: ACM TAAS. doi: 10.1145/1342171.1342172.

Virtualization



Virtualization



Cloud level

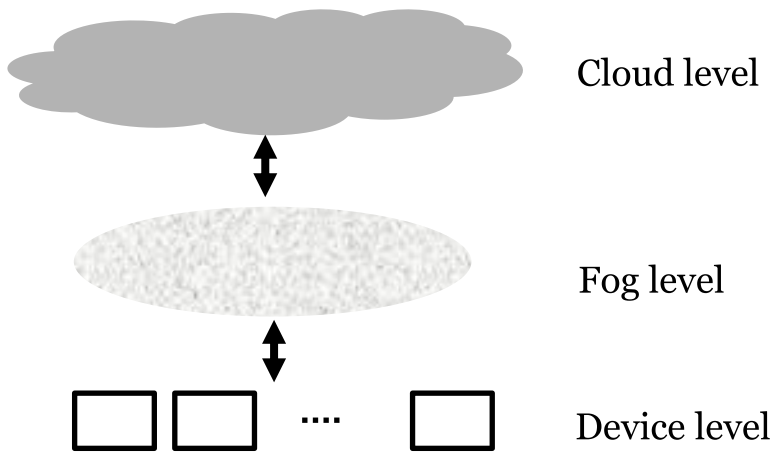
Fog level

Device level

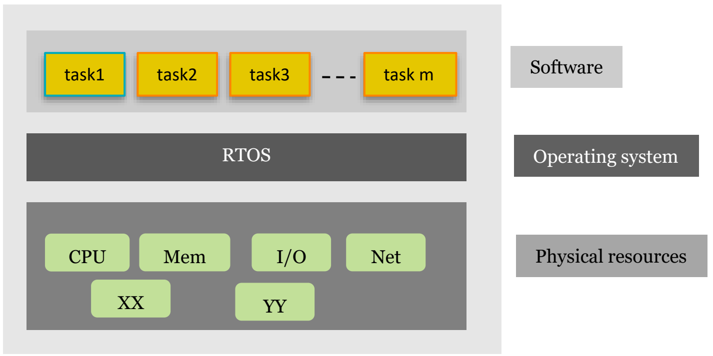
Physical processes

Virtualization in cloud

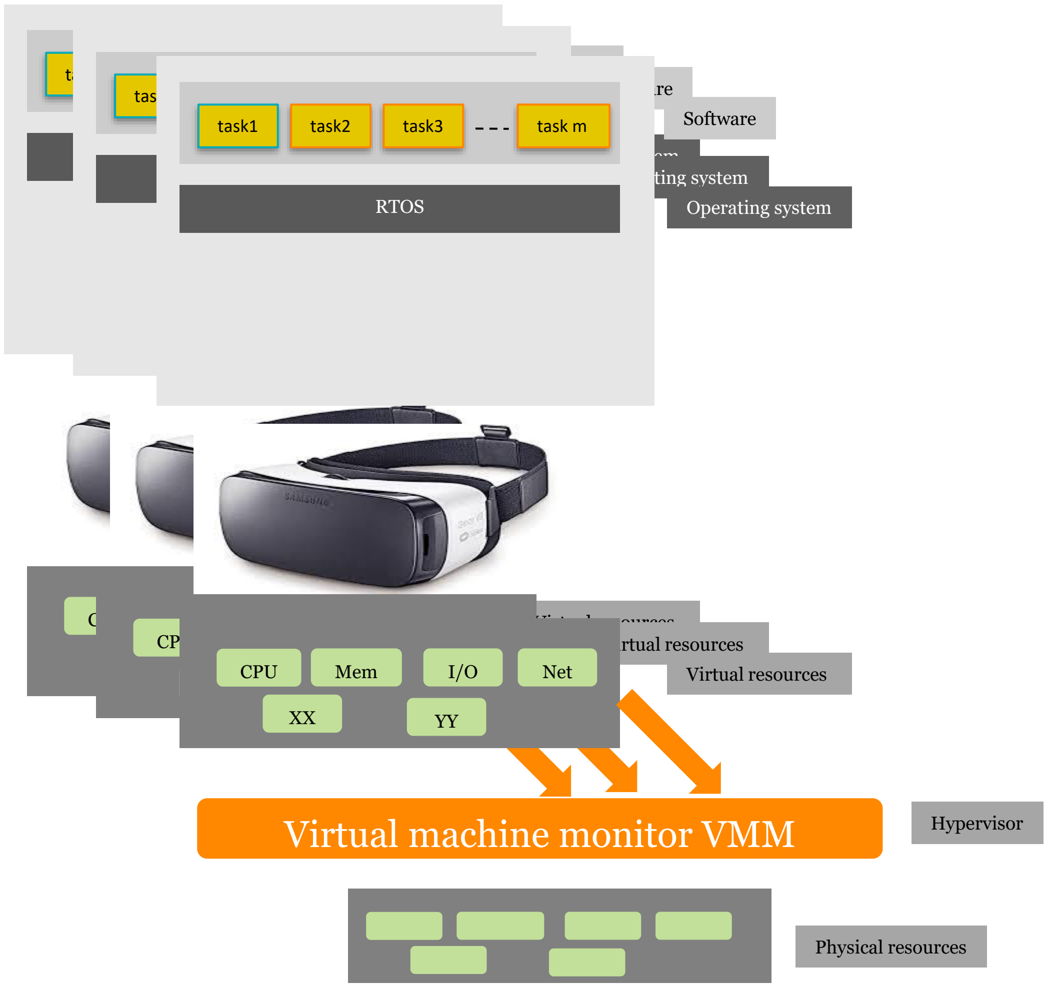
- Environment:
  - Heterogeneous resources, computation and communications
  - Dynamic nature, resources and workload
  - Sharing resources
- Requirement:
  - High Performance
  - Predictable



Challenge: How to develop and execute real-time embedded applications in such environment?

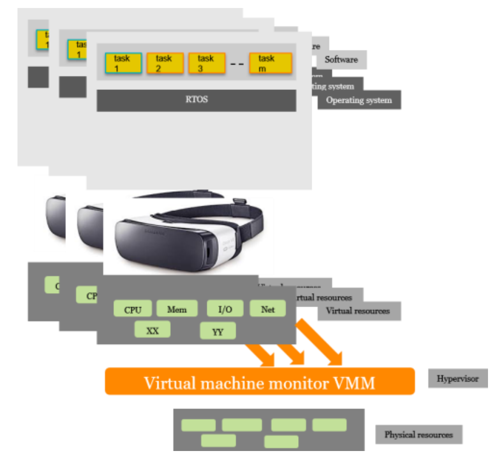


Virtualization



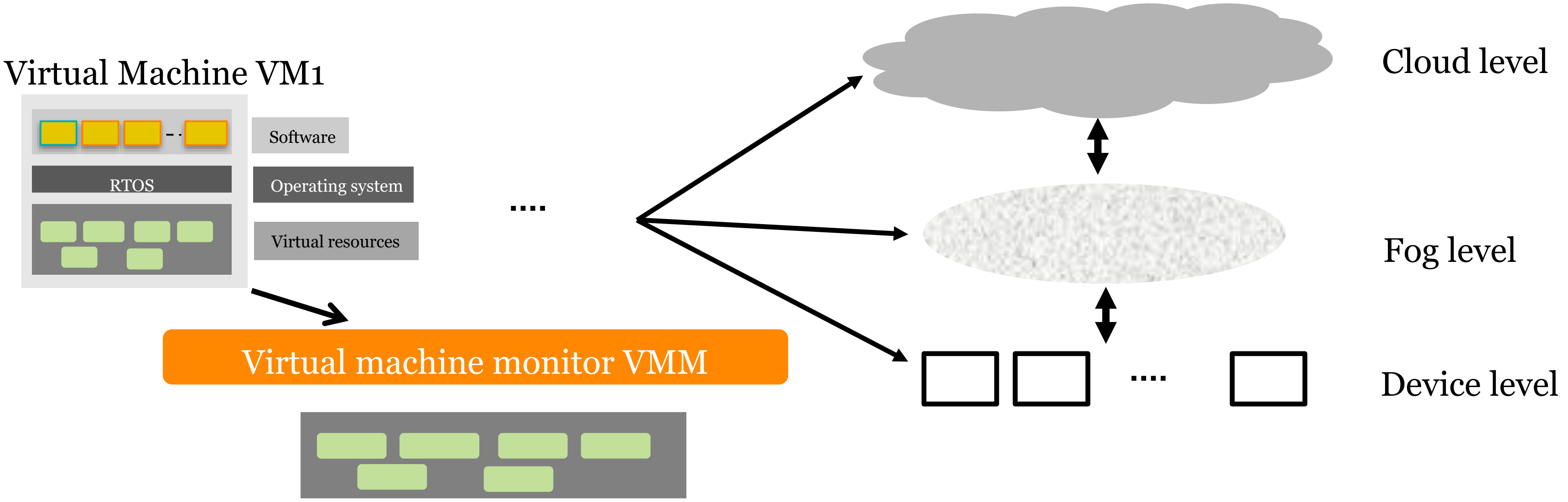
Virtualization

- Applications are executed in virtual machines
- Virtual machines provide virtual resources and access to virtual resources is managed by guest OS
- Applications are executed as if they have execlusive access to resources
- Virtual machine monitor VMM/hypervisor will map virtual resources to physical resources. i.e., Hypervisor will provide physical resources to VMs
- Scheduling algorithms are used to provide resources
- Hypervisors do not know the internal details of VMs, workload, tasks, etc.



Virtualization

- Virtualization can be applied in device, Fog and Cloud levels.



Virtualization

- Advantages:
  - Reusability (migration) of Legacy applications
  - Server consolidation
  - Faults isolation
  - Independent development of applications
  - Certification (e.g., DO-178B for avionic systems)
- Disadvantages:
  - Runtime overhead
  - Complexity

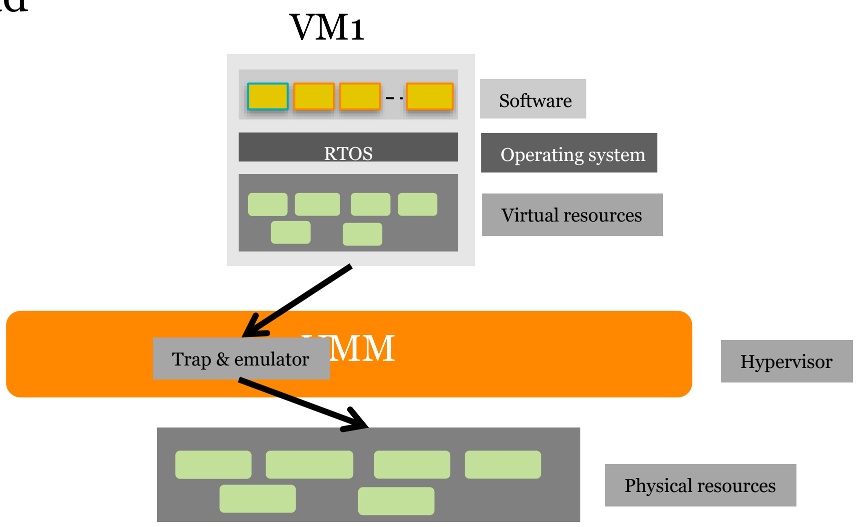
Types of virtualization

- **Full virtualization** (for example KVM)
  - Emulates full HW interfaces, BIOS, etc
  - No guest OS modification
- **Paravirtualization** (for example Xen)
  - Emulate some interfaces
    - No emulation of the processor
  - Guest OS modifications, but native guest apps
- **Container** virtualization (for example Docker)
  - Typically better performance than VMs
  - Lack of isolation from the host OS
  - More exposed to security threats

machine containers from Ubuntu

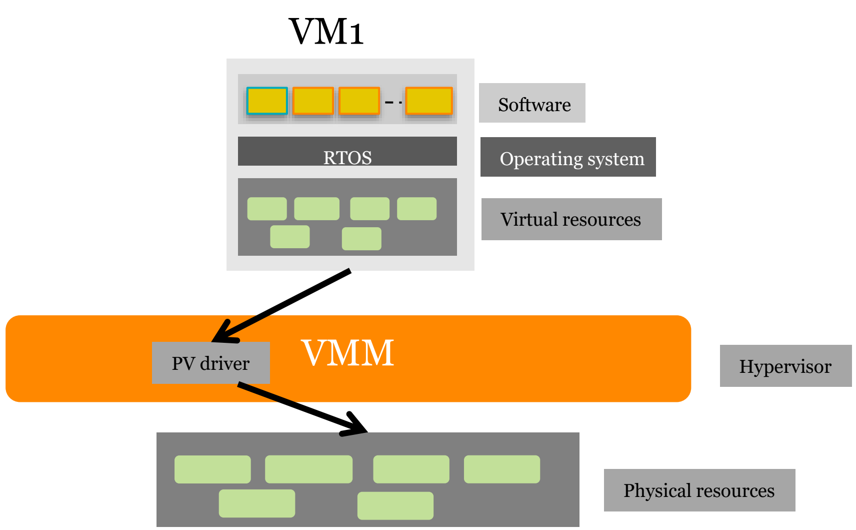
Full Virtualization

- Full Virtualization:
  - Unmodified Guest OS
  - VMMs interprets all sensitive calls from guest OSs that try to access system resources directly
  - Suitable for legacy systems
  - Runtime overhead



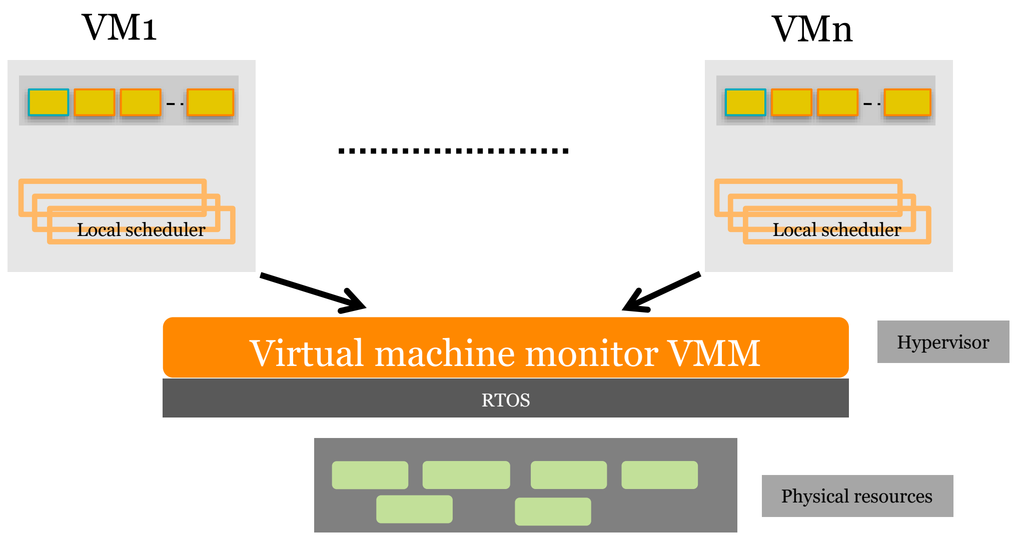
Para Virtualization

- Para Virtualization
  - Guest OS should be modified
  - Special hyper-calls to VMM are used to access system resources
  - Lower runtime overhead than the full virtualization



Containers

- Contianers
  - No OS in VMs
  - Perfromance vs flexibility vs isolation



Hardware support

- Intel VT-x for IA-32 and Intel 64:
  - provides the basic framework that virtual machine monitors (VMMs) need to operate efficiently. Privilege mode for VMM, memory protection.
- Intel VT-d for Directed I/O:
  - Makes direct access to a PCI device possible for guest systems with the help of the Input/Output Memory Management Unit. This allows a LAN card to be dedicated to a guest system.

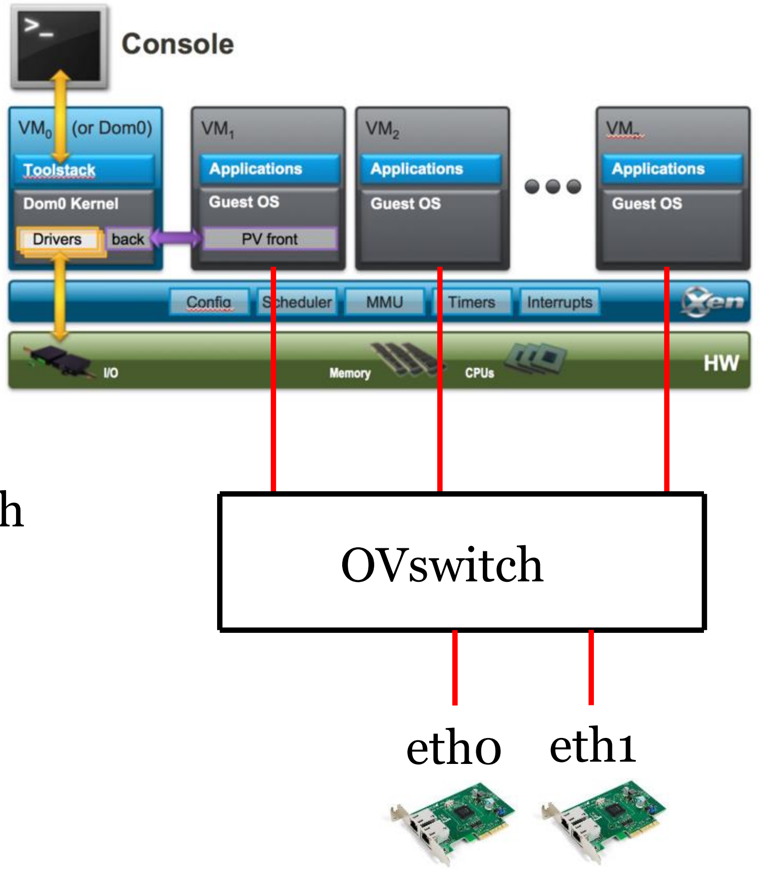
Hardware support

- Intel VT-c for Connectivity:
  - Intel I/O Acceleration Technology for the Reduction of CPU Loads
  - Virtual Machine Device Queues for the reduction of system latency
  - Single Root I/O Virtualization for the improvement of network I/O throughput

Hypervisor example

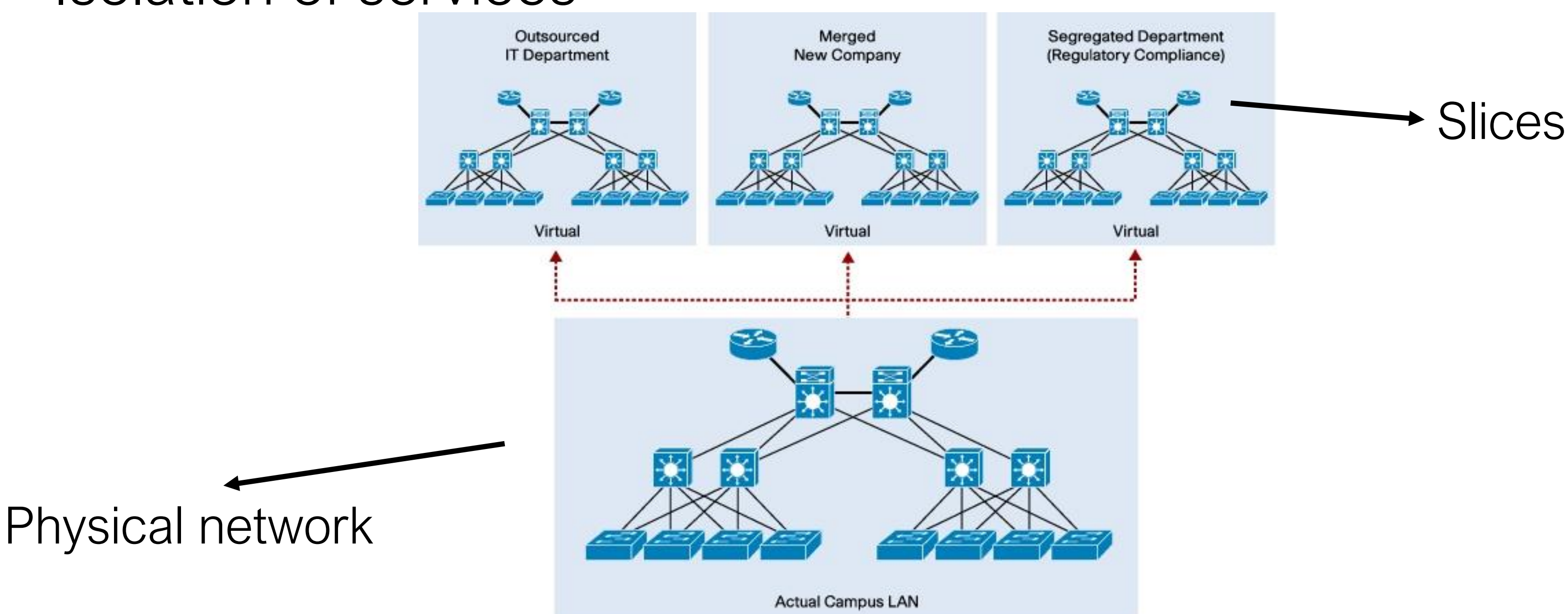
Xen

- Manage CPU, memory and interrupts
- Control Domain: has special privikeges (H/W, I/O, other VMs)
- Full- and para-Virtualization
- Toolstack: Linux
- Communication: open virtual switch can be used
- Scheduler: credit, ARINC635, RT-XEN RT schedulers



Network Virtualization

- Inherited from virtualization in computing
- The intention is to partition a physical network into several virtual networks, also known as *slices*.
- Isolation of services



Network Virtualization

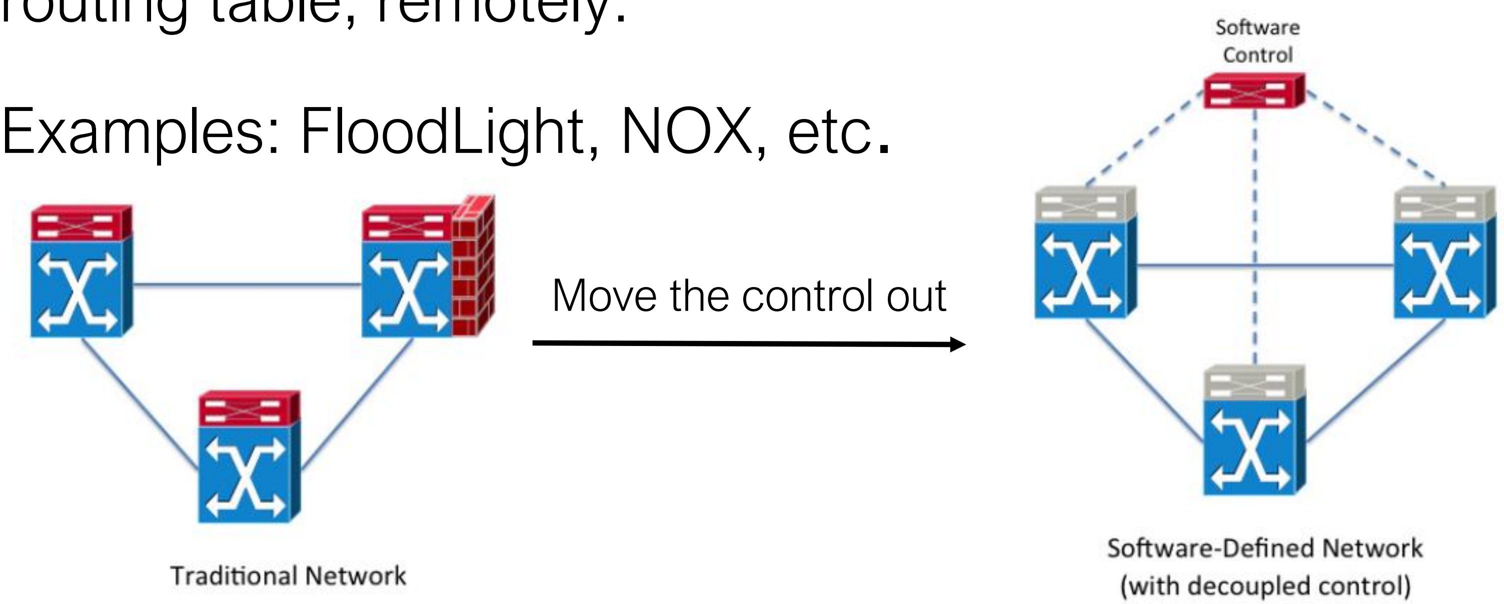
- How to obtain network virtualization?
  - With the help of
    - Network hypervisor and
    - Software Defined Networking (SDN)

Hypervisors

- Hypervisors act the same way as computer hypervisors
- They abstract the network resources, such as network link resource.
- Examples are FlowVisor, Slice Isolator, AutoSlice.

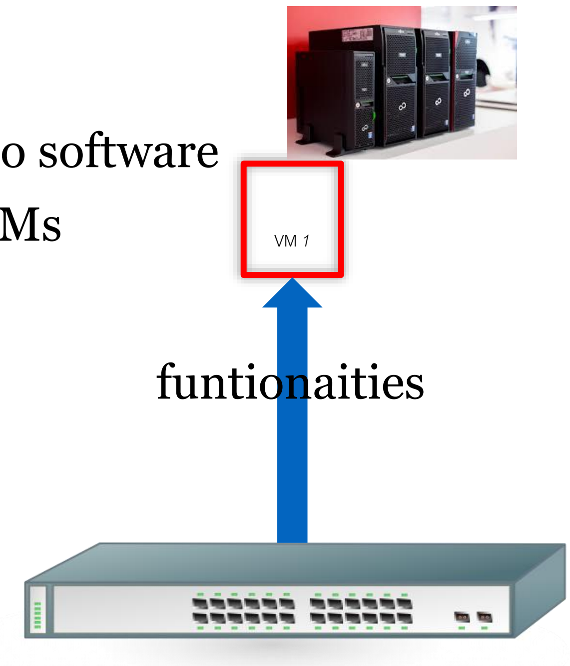
SDN

- Software Defined Networking (SDN) is a way to disintegrate the network control from network devices.
- The software control is called *SDN controller*.
- SDN controller can configure the switch and change the routing table, remotely.
- Examples: FloodLight, NOX, etc.



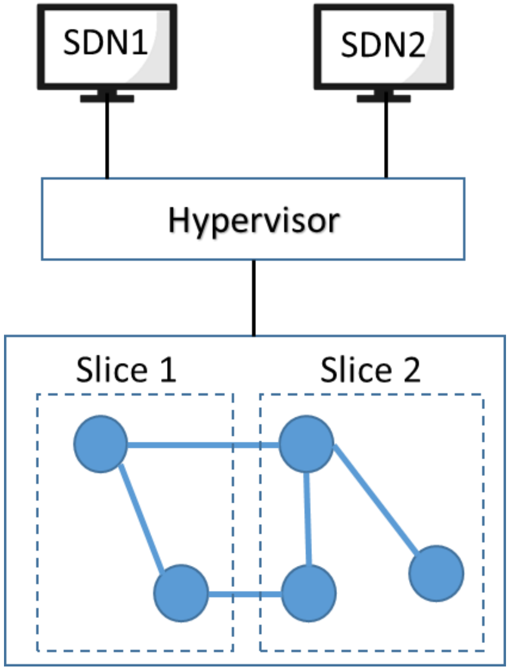
Virtual Network function VNF

- Moves network functionality from H/W to software
- VNFs implement network functions on VMs
  - Firewalling
  - Domain name service (DNS)
  - Network address translation (NAT)
  - Routing tables



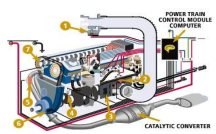
Virtualization

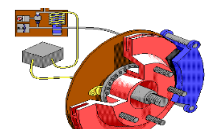
- With the help of SDN and Hypervisor, we can obtain network virtualization
  - Hypervisor creates slices
  - Each SDN can see a part of physical network, i.e., a slice
  - SDN1 controls Slice 1, SDN2 controls Slice 2.



Embedded systems and Virtualization

- Hypervisor should provide enough resources to each VM





Virtual Machine VM1

- CPU
- Memory
- Communications
- Logical resources
- I/O
- .
- .

....

Virtual Machine VMn

- CPU
- Memory
- Communications
- Logical resources
- I/O
- .
- .

Virtual machine monitor VMM

Hypervisor

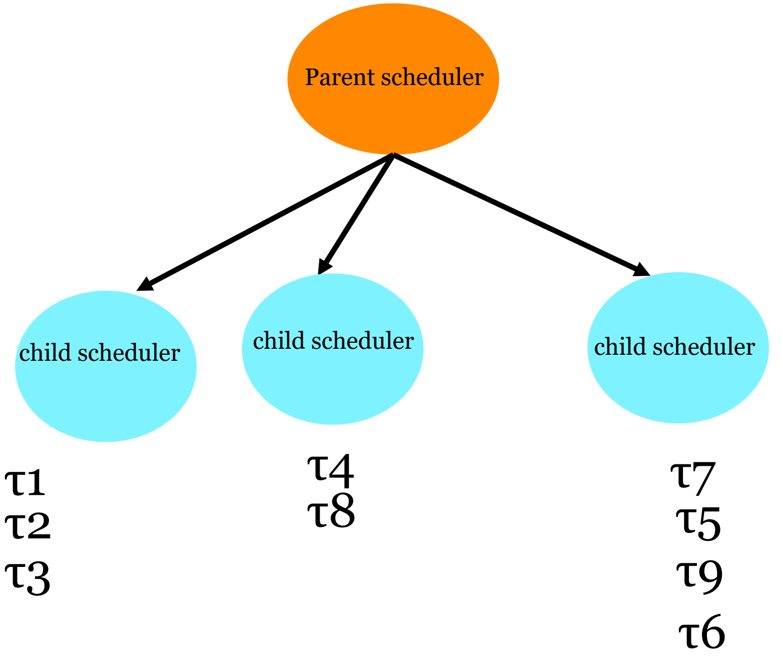
Physical resources

Scheduling problem

Real-time + Virtualization

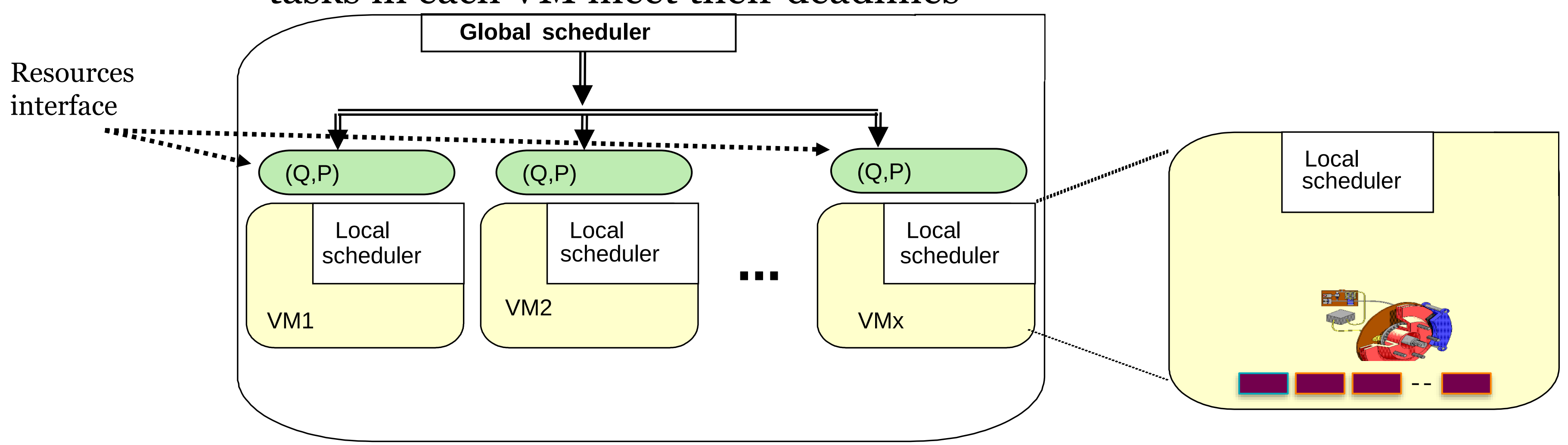
- Compute resources needed by a VM
- Proper model to express resources
- Timing verification
- Run-time support (VMM) to control resource access

Hierarchical Scheduling

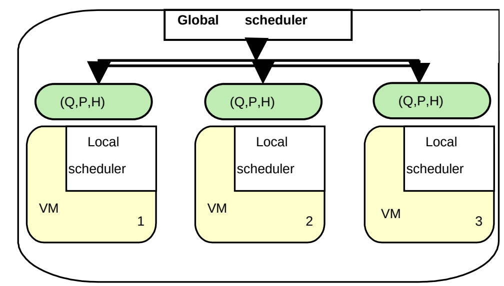


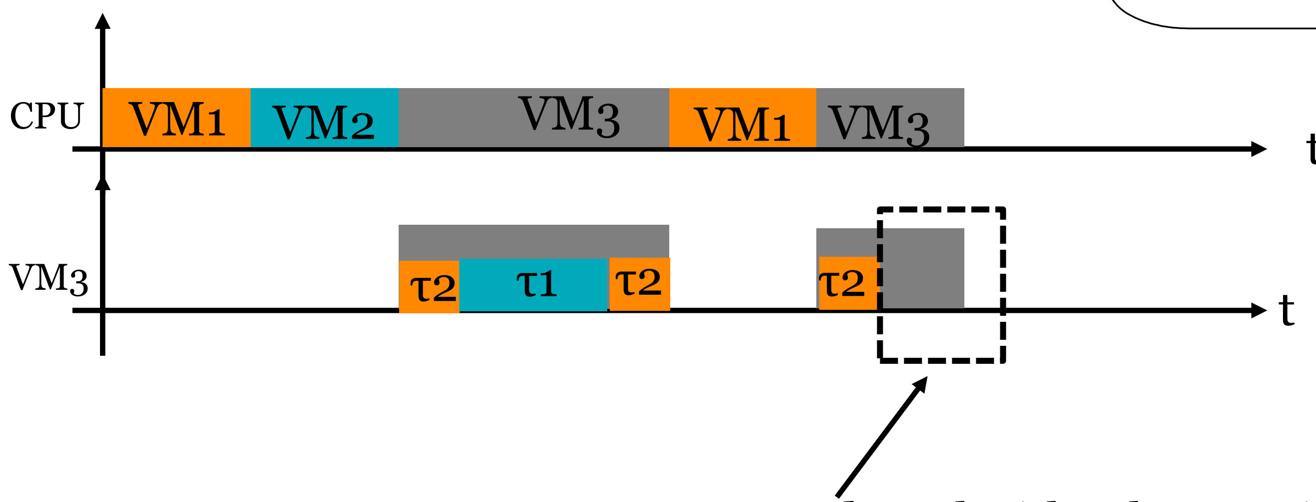
CPU

- The resource requirements by each VM is abstracted using an interface e.g., (Q,P), where Q is budget provided every period P
- Based on the interface the global/system level scheduler (hypervisor) will select one VM. Once a VM is selected its local scheduler will select which task to execute.
- RT requirements, all VMs meet their deadlines and all tasks in each VM meet their deadlines



Example





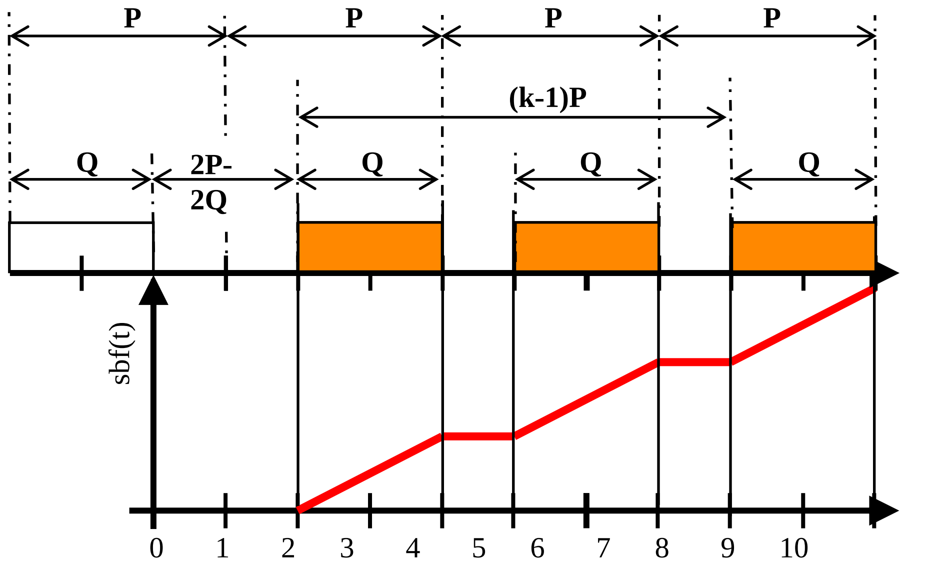
No ready task either busy wait or suspend. What are the advantages and disadvantages of each choice?

CPU

- Periodic resource models, predefined budget will be serverd periodically
- Minimum resource supply *sbf*, when the task request is ready just after the budget depletion and the budget was served early and all future budget will be severed late

$$\mathtt{sbf}_\Gamma(t, \mathsf{BD}) = \begin{cases} t - (k-1)(P-Q) - BD & \text{if } t \in W^{(k)} \\ (k-1)Q & \text{otherwise,} \end{cases} \quad (1)$$

where $k = \max\left(\lceil (t + (P-Q) - BD)/P \rceil, 1\right)$ and $W^{(k)}$ denotes an interval $[(k-1)P + \mathsf{BD}, (k-1)P + \mathsf{BD} + Q]$.



CPU

- Maximum resource request rbf, Example considering fixed priority scheduling local scheduler

$$\mathtt{rbf}_{\mathsf{FP}}(i,t) = C_i + \sum_{\tau_k \in \mathrm{HP}(i)} \left\lceil \frac{t}{T_k} \right\rceil \cdot C_k,$$

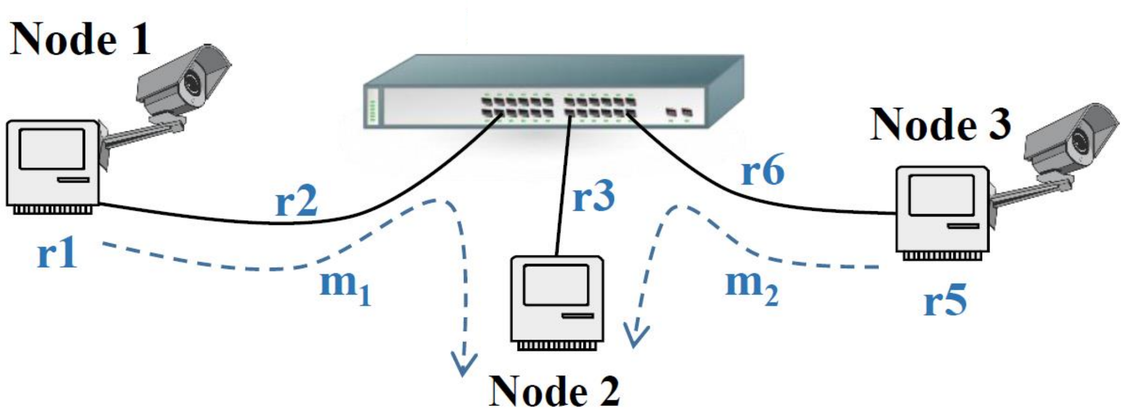
Where C is execution time, T is period and HP is the set of higher priority tasks

- Schedulability test: system is schedulable if the maximum resource request <= minimum resource supply.

$$\forall \tau_i, 0 < \exists t \le D_i \ \ \mathtt{rbf}_{\mathsf{FP}}(i,t) \qquad \le \mathtt{sbf}(t),$$

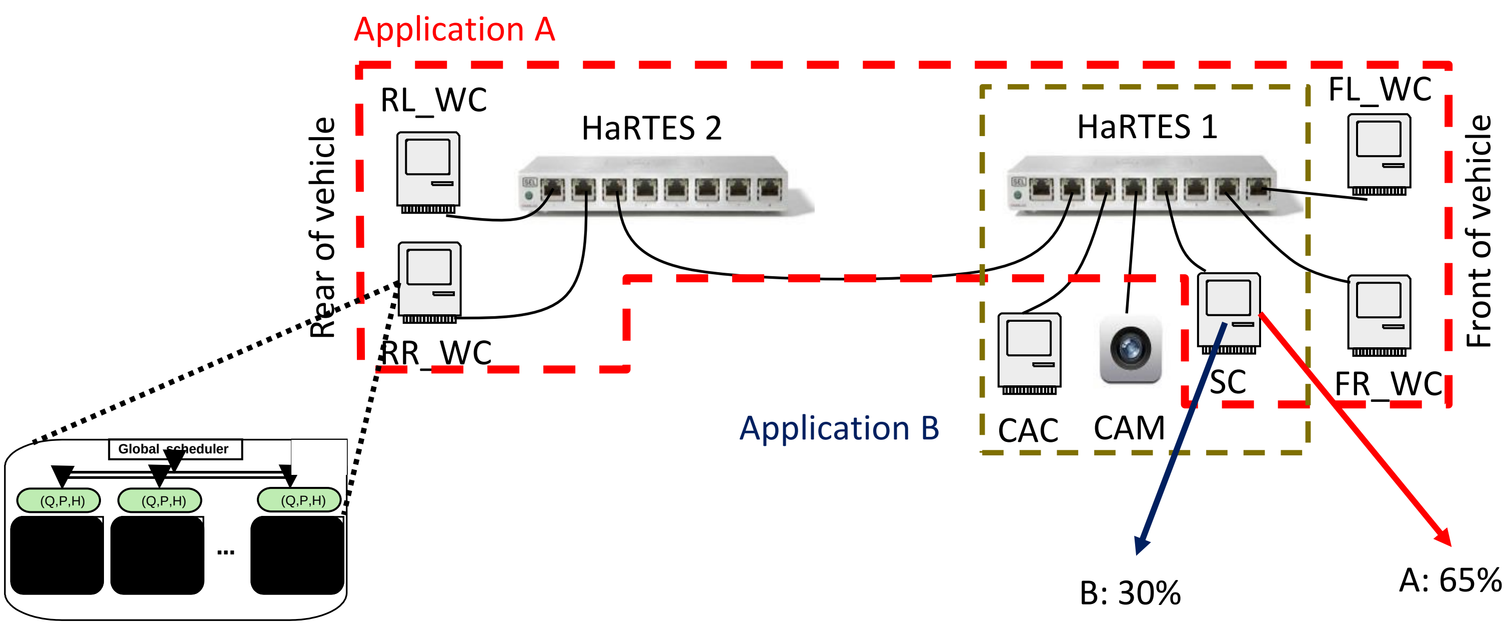
Communication resource

- Messages share netwrok channels
- RT predictability: messages need to be scheduled
- Assign bandwidth to each message/group of messages, budget every period (trafic shaping)



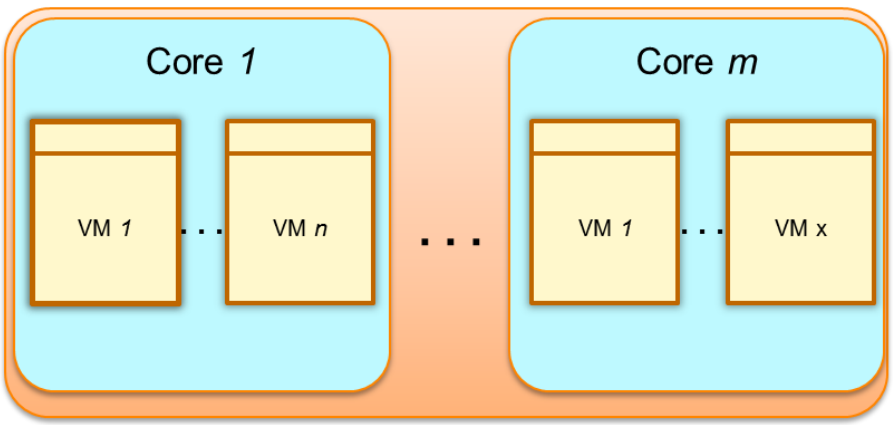
Distributed systems CPU+Network

- End to End model
- Budget for each resource



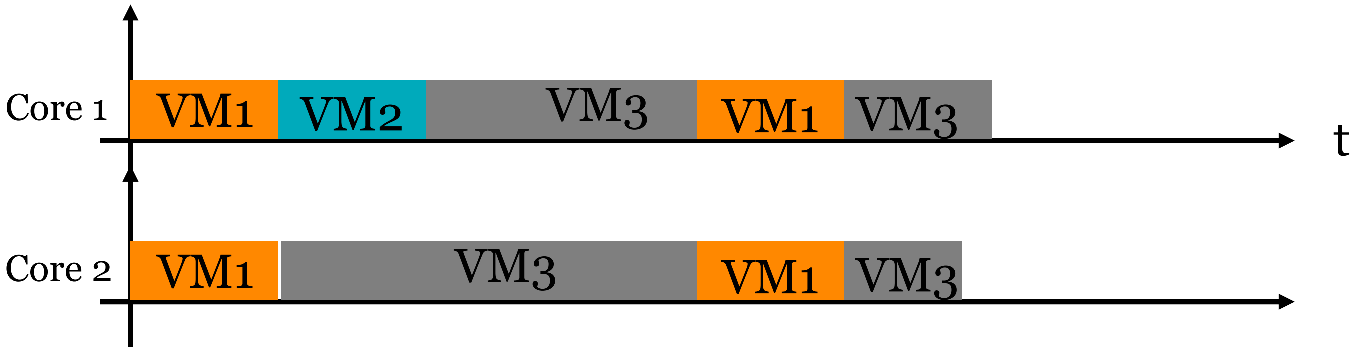
MultiCore

- Virtual machines can execute on more than one core
- Global scheduling or partitioned scheduling for VM and tasks within each VM.



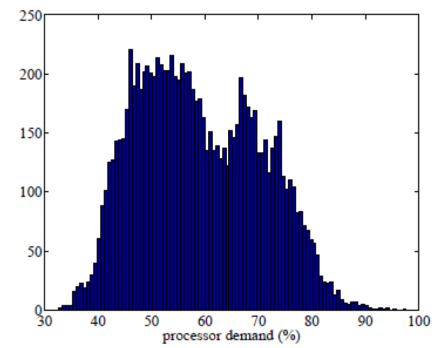
MultiCore

- Sequential tasks can not be parallilized

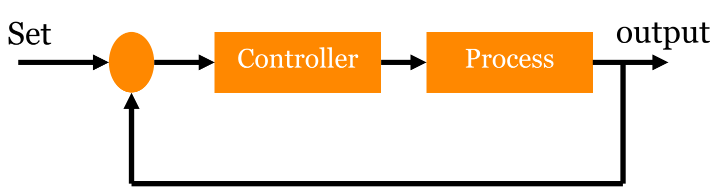


Dynamic Systems

- Resource requirements for certain software applications might be dynamic (changing)
- Resource reservation
  - Worst-case resource: wasting resources
  - Average case: too many time violations
  - Feedback control
    - Controlled parameters
    - Design of the controller
    - Control model of the system



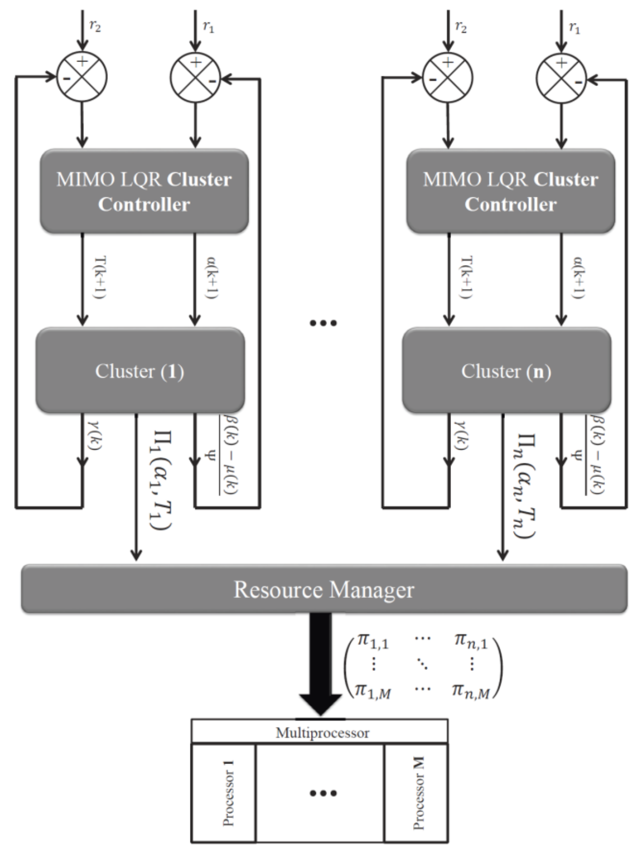
The distribution of processor demand percentage of a video decoder



$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k),$$

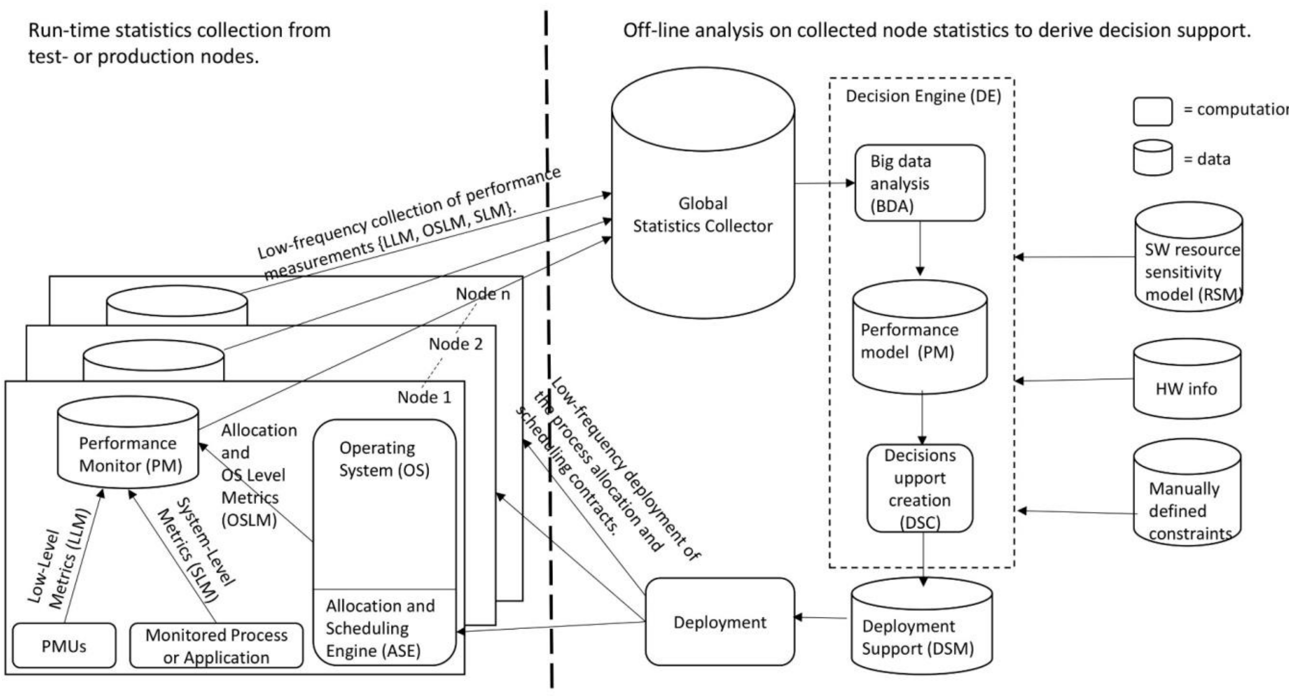
Feedback scheduling

- Soft real time
- Feedback control
  - Controlled parameters: VM budget, VM period, VM allocation
  - O/P: deadline miss ratio/time
  - LQR controller



Feedback scheduling

- Control of budget and period and VM allocation
- Controller using AI
- Measure low level parameters, cache, memory, bandwidth, ... using perfromance counters



Other challenges

- Large scales systems
- Security and safety
- Services allocation and migration
- Service Level Agreement SLA