Resource Allocation in Cloud and Fog Computing

Fog and Cloud Computing

Francescomaria Faticanti, Inria Sophia Antipolis, France

May 24, 2022

francescomaria.faticanti@inria.fr

Why Resource Allocation?

Several benefits

- Save money
- Improve productivity
- Saving time
- Effective use of resources

Why Resource Allocation?

Several benefits

- Save money
- Improve productivity
- Saving time
- Effective use of resources

How hard is that?

Resource Allocation Problem

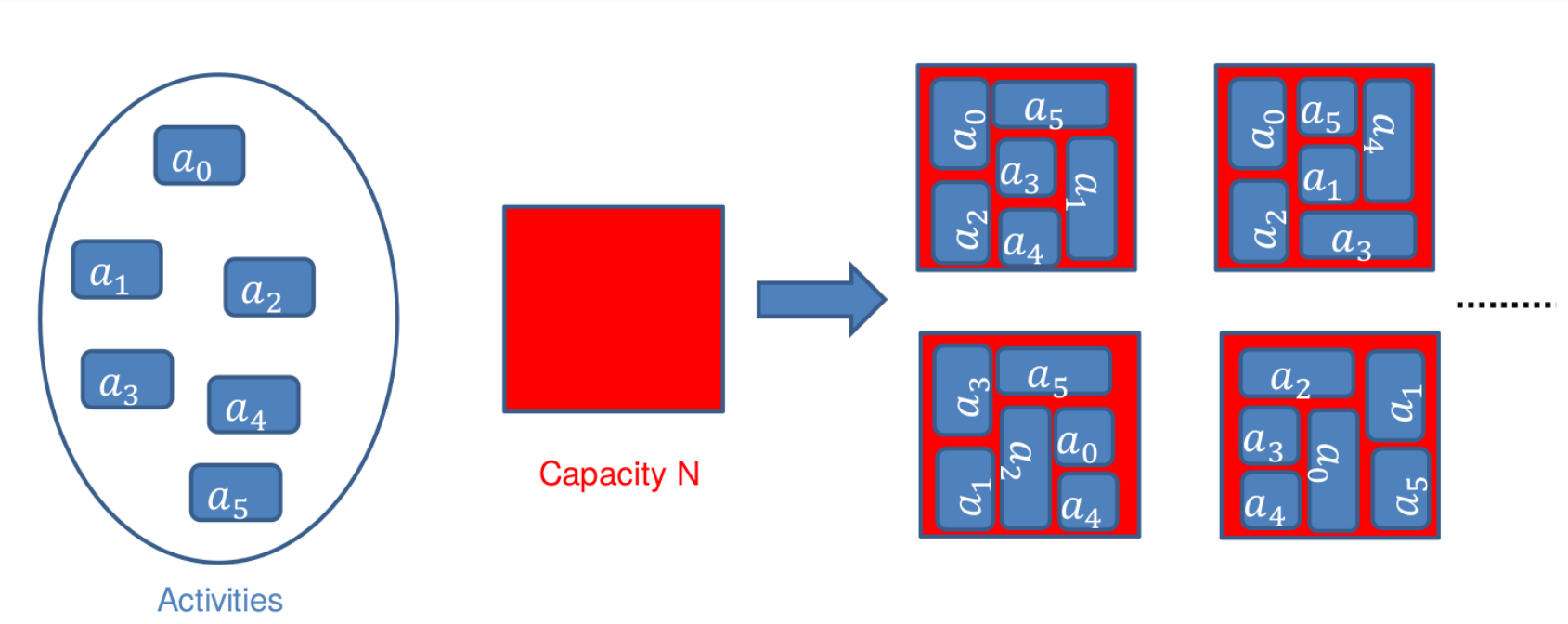
*G*iven a type of resource whose total amount is equal to N, we want to allocate it to n activities so that a certain **objective function** is minimised (or maximised)

Resource Allocation Problem

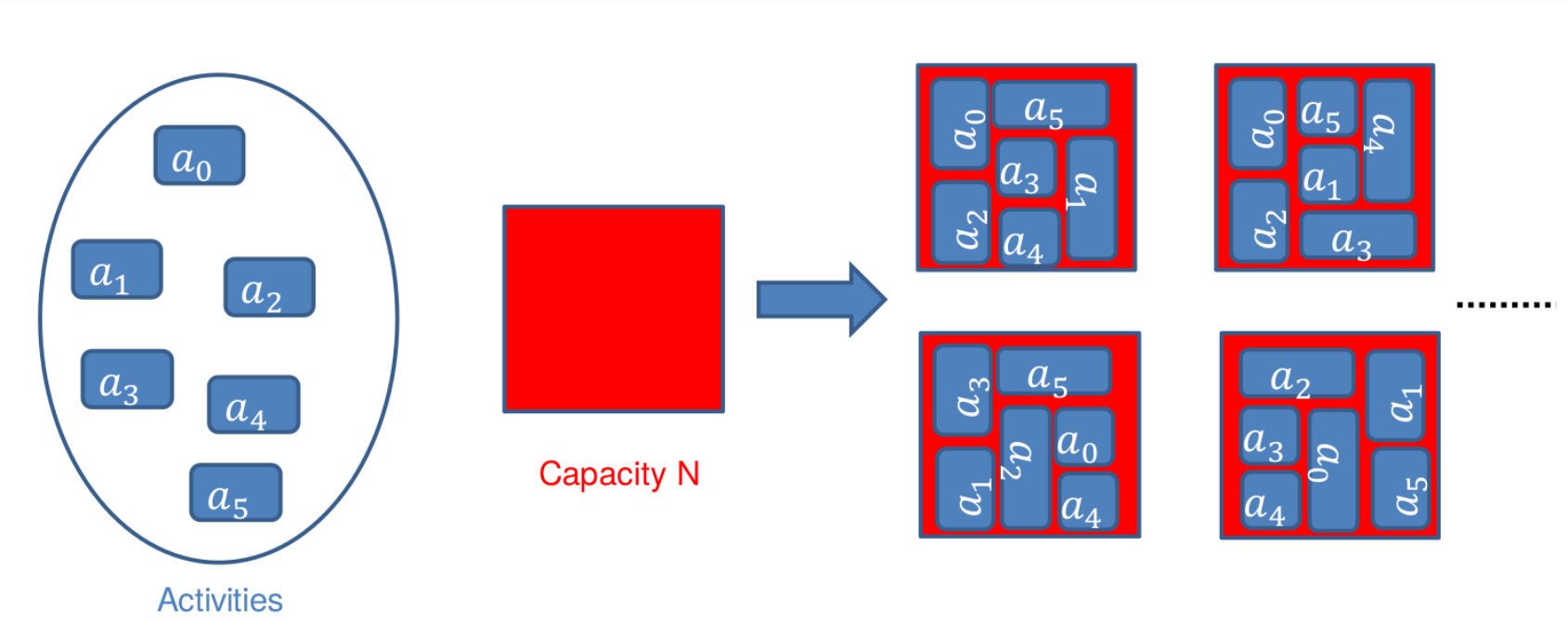
*G*iven a type of resource whose total amount is equal to N, we want to allocate it to n activities so that a certain **objective function** is minimised (or maximised)

- **Objective function**: cost or loss (profit or reward) of the resulting allocation

Resource Allocation Problem (1)



Resource Allocation Problem (1)



Each configuration: specific objective value

Formally

$$minimize\ f(x_1, x_2, \ldots, x_n)$$

$$s.t.$$

$$(1)\ \sum_{\{j=1\}}^{n} x_j = N,$$

$$(2)\quad x_j \geq 0,\quad j = 1,2,\ldots,n.$$

Objective function: its value depends on the values assigned to the variables

Constraint 1: the total amount reserved to all the activities must be equal to the available resource

Constraint 2: the amount reserved to each activity must be nonnegative

Examples

Distribution of search effort:

- Detect the position of an object whose possible positions are among those numbered from 1 to n
- The probability of the object being in position j is p_j
- $(1 - e^{-\alpha x_j})\, p_j$: conditional probability of detecting the object in position j
- x_j: amount of search effort
- **Constraint**: N is the total amount of effort
- **Objective**: maximize the overall probability of detecting the object

Examples

Resource Distribution Problem:

- n locations to which resources such as $n$ewspapers are distributed from the central factory
- Demand at each location j
- $q_j(x_j)$: the expected cost at location j when x_j is allocated to j
- N: total amount of resources allocated to all locations

$$minimize \sum_{\{j=1\}}^{n} q_j(x_j)$$

$$subject\ to \sum_{\{j=1\}}^{n} x_j = N,$$

$$x_j \geq 0.$$

Examples

Online Caching Problem

- Requests from a catalog of N files $\mathscr{N} = \{1, \ldots, N\}$
- $w_i \in \mathbb{R}_+$: cost to serve a request for file i by a remote server
- Single-cache system with capacity K
- Storage: arbitrary fractions of files
- Slotted time $t \in \mathscr{T} = \{1, \ldots, T\}$
- Cache state $\mathbf{x}_t = [x_{t,i}]_{i \in \mathscr{N}}$ drawn from

$$\mathscr{X} = \left\{ \mathbf{x} \in [0,1]^N : \sum_{i=1}^{N} x_i = K \right\},$$

where $x_{i,t}$: fraction of file i stored at time t

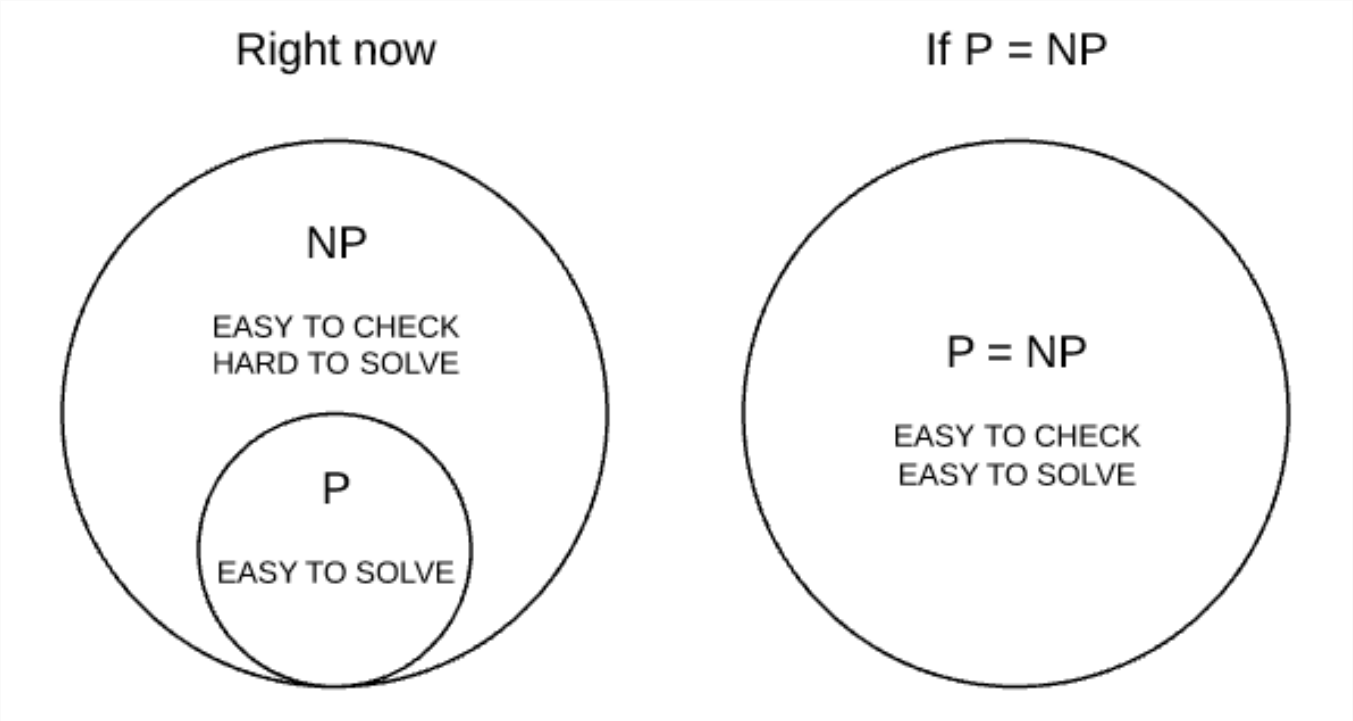
Online Caching Problem (1)

$$\min_{\mathbf{x}\in\mathscr{X}} \sum_{t=1}^{T} f_t(\mathbf{x}) \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{N} x_{t,i} = K, \quad \forall t \in \mathscr{T} \tag{2}$$

$$0 \leq x_{t,i} \leq 1, \quad \forall i \in \mathscr{N}, \forall t \in \mathscr{T} \tag{3}$$

P vs NP



Millennium Prize Problem

Decision Problem

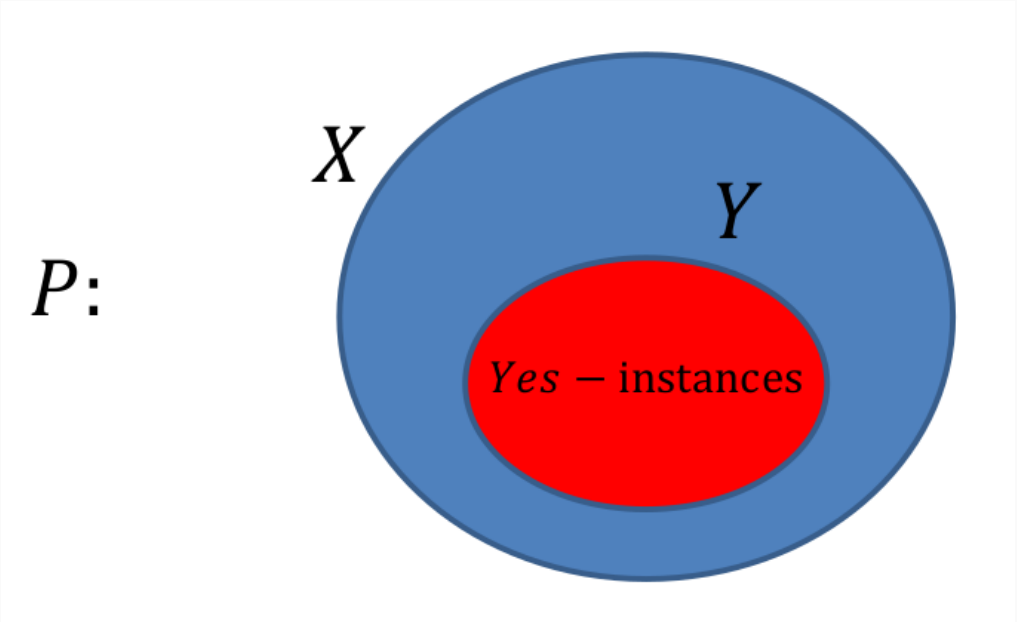
Definition (Decision Problem)

A **decision** problem is a pair $P = (X, Y)$, where X is a language decidable in polynomial time and $Y \subseteq X$. Elements of X: **instances** of P; elements of Y: **yes-instances**.

Decision Problem

Definition (Decision Problem)

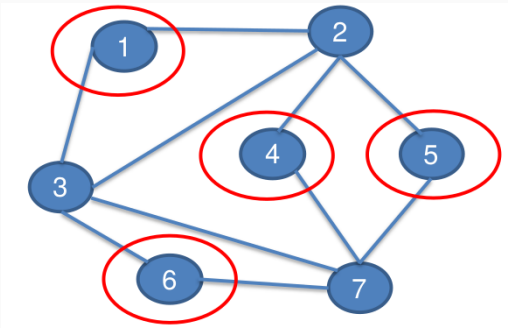
A **decision** problem is a pair $P = (X, Y)$, where X is a language decidable in polynomial time and $Y \subseteq X$. Elements of X: **instances** of P; elements of Y: **yes-instances**.



Example (Decision Problem)

Problem (Independent Set)

Given a graph G, a k-independent set is a subset of its vertices of cardinality k, where no couple of vertices is connected by an edge in G.



- Instances: $\langle G, k \rangle$
- Yes-instances: $\langle G, k \rangle$ s.t. G contains an independent set of k vertices

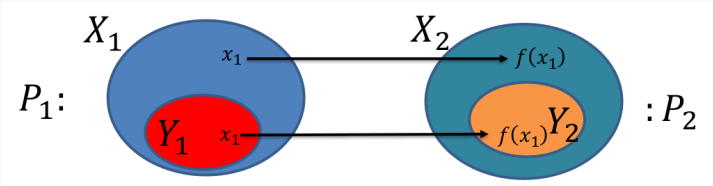
4-independent set

Polynomial Reduction

Definition (Polynomial Transformation)

Let $P_1 = (X_1, Y_1)$ and $P_2 = (X_2, Y_2)$ be decision problems. P_1 **polynomially transforms** to P_2 ($P_1 \leq_p P_2$) if there is a function $f : X_1 \to X_2$, computable in polynomial time, s.t.

$$f(x_1) \in Y_2, \forall x_1 \in Y_1 \wedge f(x_1) \in X_2 \setminus Y_2, \forall x_1 \in X_1 \setminus Y_1.$$



Proposition

If $P_1 \leq_p P_2$ and there is a polynomial time algorithm for P_2, then there is a polynomial time algorithm for P_1.

NP-completeness

Definition

A problem P is **NP-complete** if

- $P \in NP$
- $\forall P' \in NP,\ P' \leq_p P$ (**NP-hardness**)

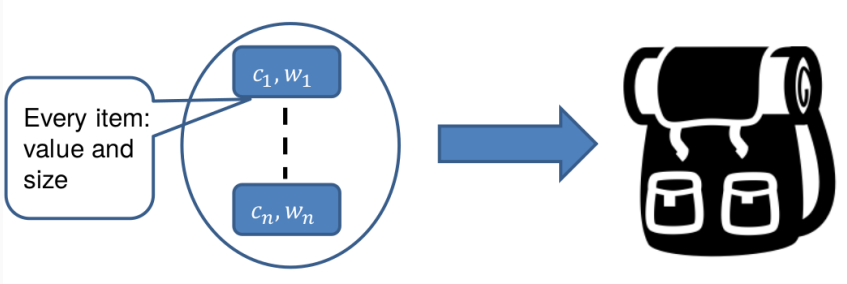
Theorem

SATISFIABILITY is NP-complete (Cook [1971])

NP-hard Problems

Knapsack:

- *Instance*: A number $n \in \mathbb{N}$ and nonnegative integers c_i, w_i and W for $i = 1, \ldots, n$.
- *Task*: Find a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} c_i$ is maximum.

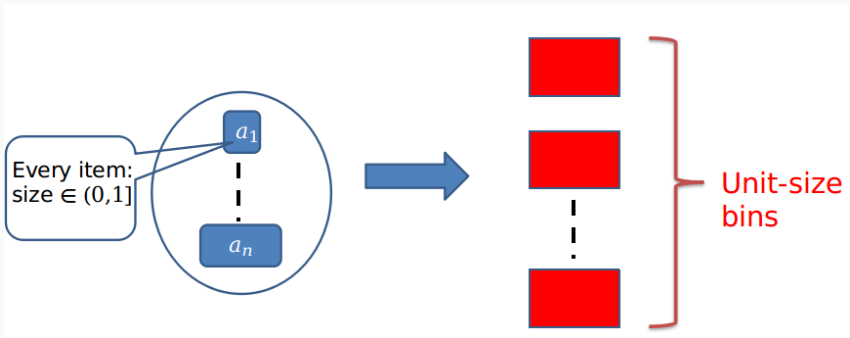


- Maximize the total value of collected items without exceeding the knapsack's capacity

NP-hard Problems

Bin-Packing:

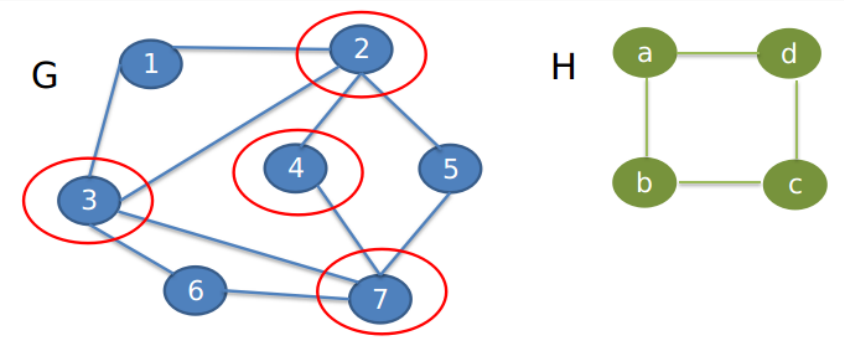
- *Instance*: A list of nonnegative numbers $a_1, \ldots, a_n \leq 1$.
- *Task*: Find a $k \in \mathbf{N}$ and an assignment $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ with $\sum_{i:f(i)=j} a_i \leq 1, \forall j \in \{1, \ldots, k\}$ s.t. k is minimum.



- Packing all the items using the minimum number of bins

Subgraph Isomorphism:

- *Instance*: Two graphs $G = (V, E)$ and $H = (V', E')$.
- *Task*: Find a subgraph $G_0 = (V_0, E_0) : V_0 \subseteq V, E_0 \subseteq E$ s.t. $G_0 \cong H$.



- Does G contains a subgraph isomorphic to H?

Challenge

- **Theory**: We cannot find poly-time algorithms for NP-hard problems (unless $P = NP$)
- **Desired features**:
  - Optimality
  - Poly-time
  - Arbitrary instances of the problem
- **Trade-off**:
  - Quality of the solution
  - Efficiency

Approximation algorithms

Approximation

Approximation algorithms

- Approximation guarantees

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case
- Efficient

Heuristics

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case
- Efficient

Heuristics

- No approximation guarantees

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case
- Efficient

Heuristics

- No approximation guarantees
- Experimental proof of approximation

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case
- Efficient

Heuristics

- No approximation guarantees
- Experimental proof of approximation
- Specific case (most of the time)

Approximation

Approximation algorithms

- Approximation guarantees
- Approximation proof
- General case
- Efficient

Heuristics

- No approximation guarantees
- Experimental proof of approximation
- Specific case (most of the time)

Note: hard to find good approximation algorithms for several NP-hard problems

Example

Online Bin-Packing Consider the items in a given order and place them one by one.
Several heuristics:

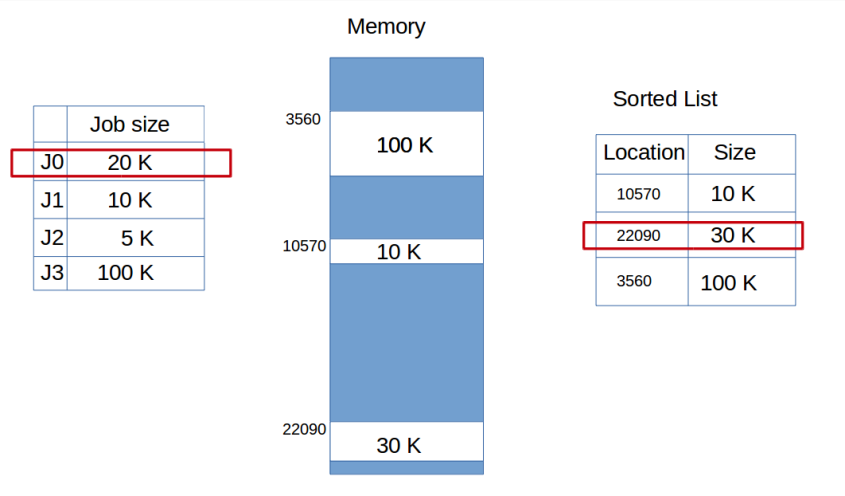
- **First-Fit**
- **Best-Fit**
- **Worst-Fit**
- **Next-Fit**

Memory Allocation (Best-Fit)

- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: It keeps the free-memory list in increasing order by size. The OS allocates the input job to the first fitting free partition in the sorted list.

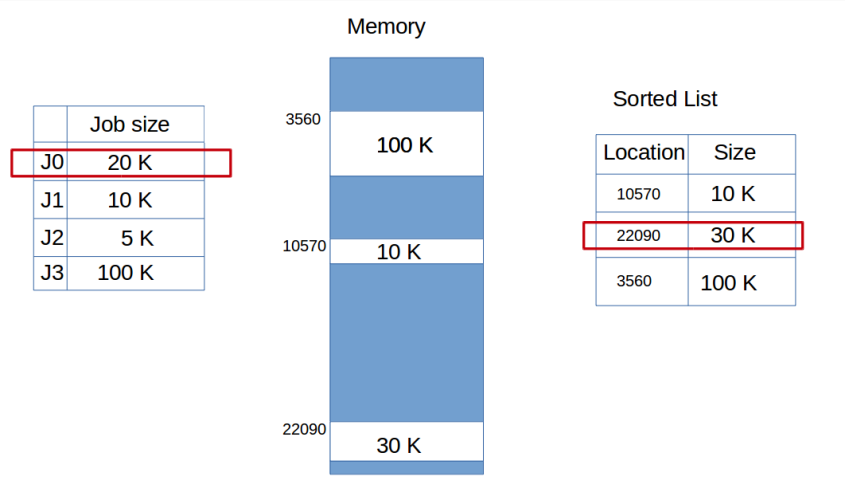
Memory Allocation (Best-Fit)

- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: It keeps the free-memory list in increasing order by size. The OS allocates the input job to the first fitting free partition in the sorted list.



Memory Allocation (Best-Fit)

- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: It keeps the free-memory list in increasing order by size. The OS allocates the input job to the first fitting free partition in the sorted list.



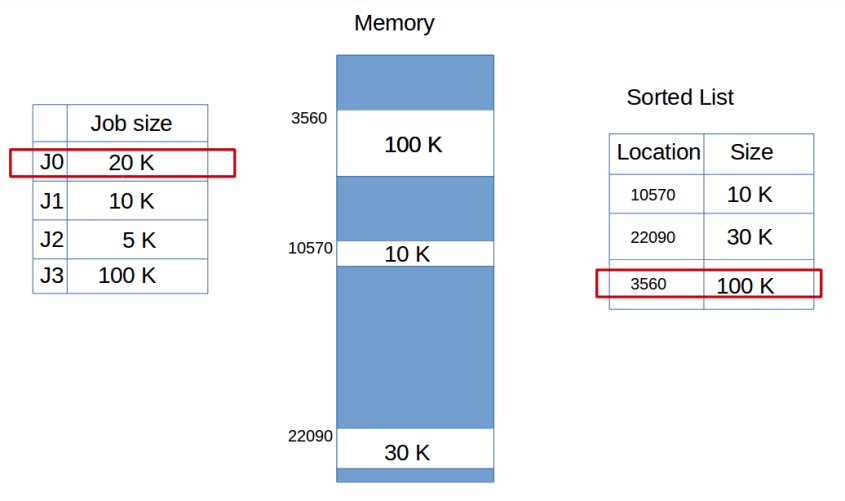
Pros and cons?

Memory Allocation (Worst-Fit)

- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: The OS allocates the input job to the first fitting free partition with the maximum size in the list.

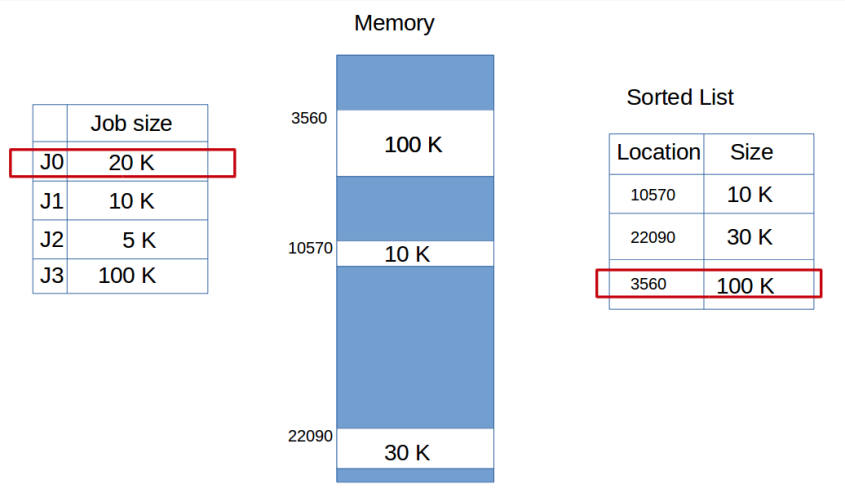
Memory Allocation (Worst-Fit)

- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: The OS allocates the input job to the first fitting free partition with the maximum size in the list.



Memory Allocation (Worst-Fit)

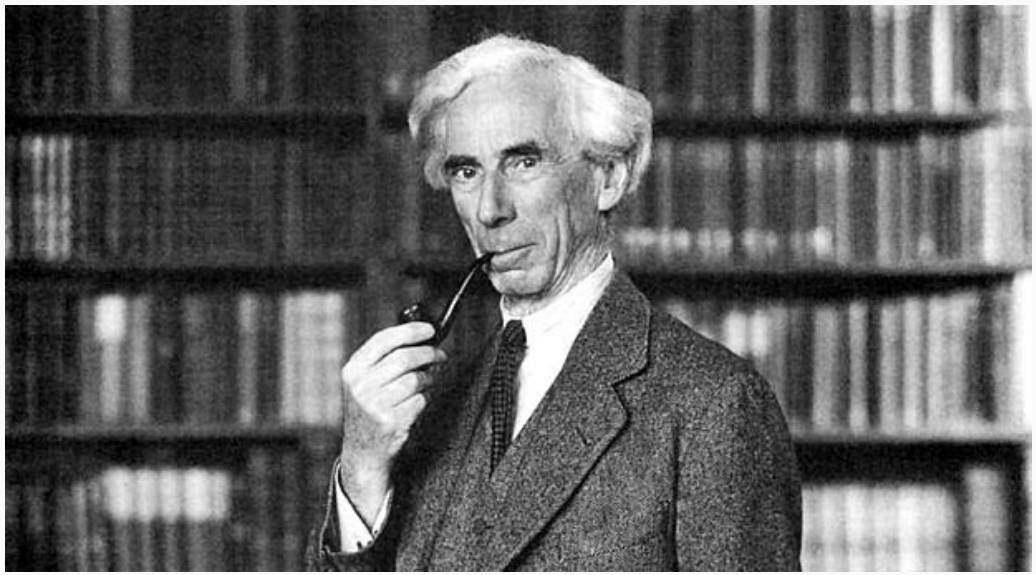
- *Input*: A fixed-size job and the list of free-memory holes.
- *Algorithm*: The OS allocates the input job to the first fitting free partition with the maximum size in the list.



Pros and cons?

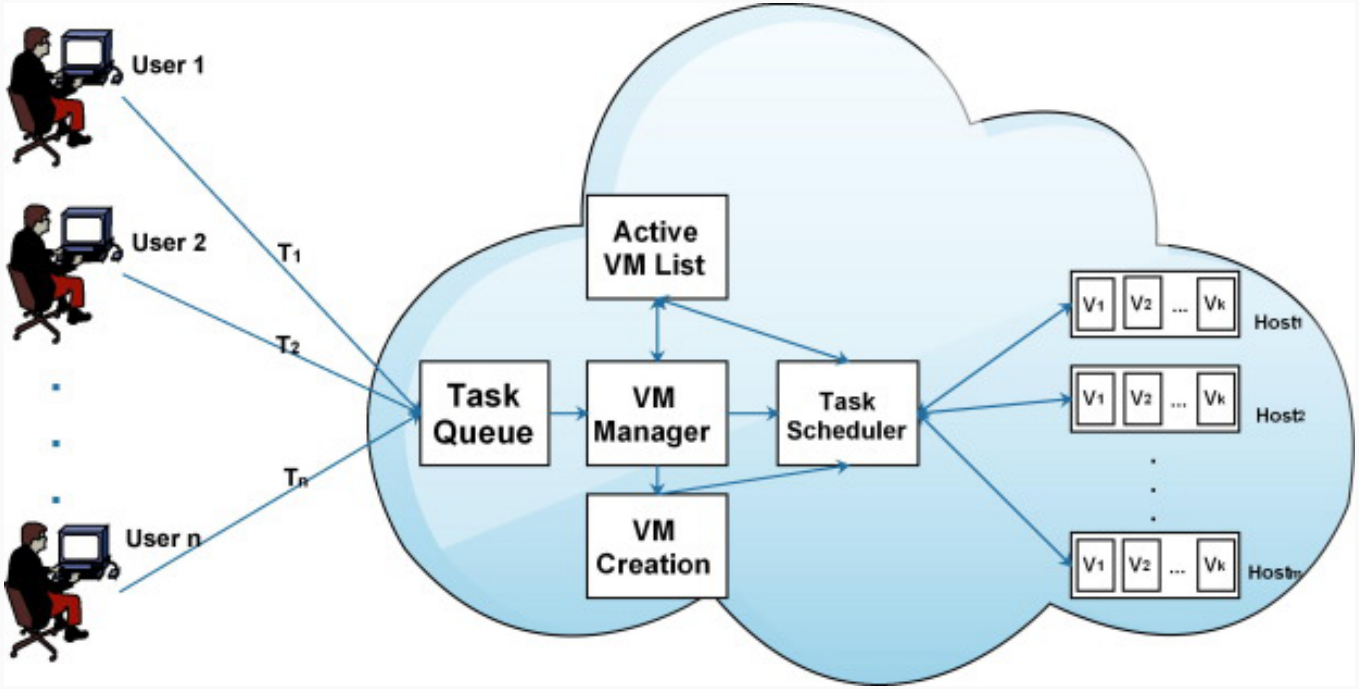
Approximation

"Although this may seem a paradox, all exact science is dominated by the idea of approximation." **Bertrand Russell**



Resource allocation in the Cloud

Cloud


User 1
User 2
User n
T1
T2
Tn
Active VM List
Task Queue
VM Manager
Task Scheduler
VM Creation
V1
V2
...
Vk
Host1
Host2
Hostm

Resource Allocation in Cloud

- Applications run on VMs
- Virtualization: splitting and assigning physical servers to VMs
- Active servers: main energy consumer in data centers
- Energy consumption: up to 50% or more of the total operational costs
- **Problem**: assigning VMs with volatile demands to physical servers such that energy costs are minimized
- **Observation**: minimizing energy costs $\rightarrow$ minimizing the number of active servers

Resource Allocation in Cloud

Typical approaches:

- Static Allocation
  - Optimization
  - Approximation
- Dynamic Allocation
  - Optimal Control techniques
  - Service live migration

Static Allocation

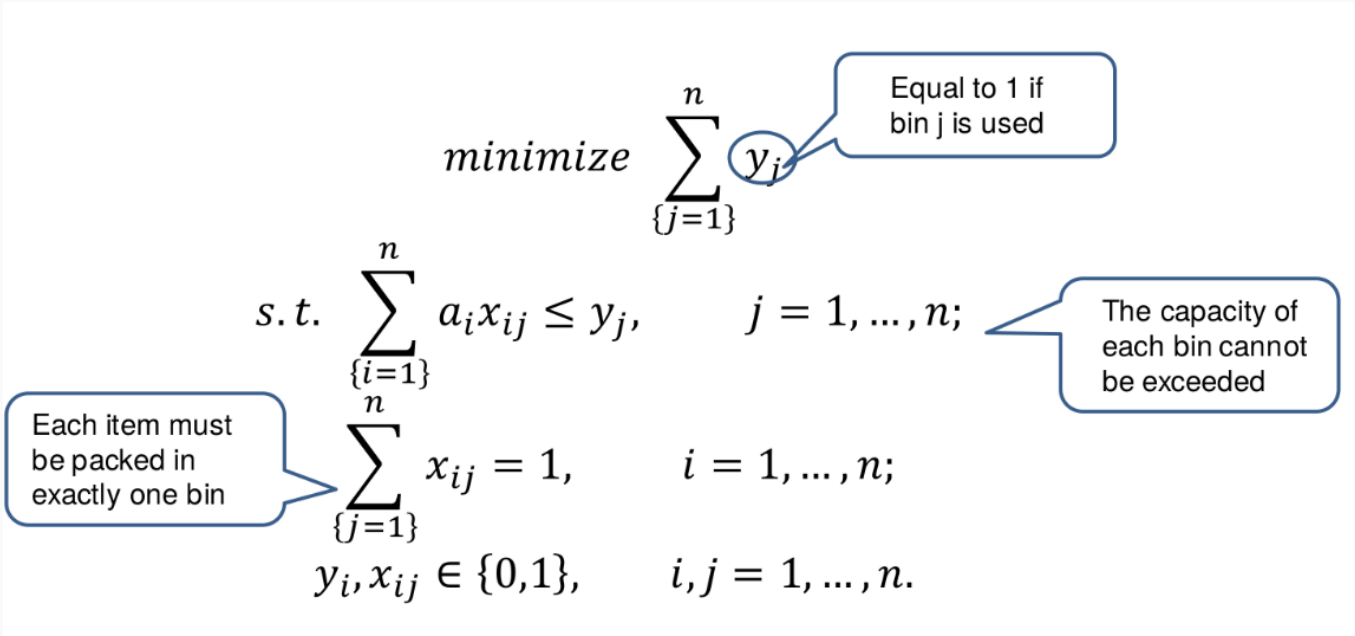
- **Assumption**: workloads patterns are known
- **Challenge**: NP-hardness
- **Two main approaches**:
  - Mathematical Programming
  - Approximated Solutions

Mathematical Programming

- **Pros**:
  - Mathematical formalization
  - Efficient solvers for problem resolution
- **Cons**:
  - Not scalable (as the instance size grows)
  - Most of the problems are NP-hard

Examples

- **Bin-packing problem**: Given n items with sizes $a_1, \ldots, a_n \in (0, 1]$, find a packing in unit-sized bins that minimizes the number of used bins.



Examples

- **Server Consolidation problem**: The IT service manager needs to consolidate servers by assigning n services to target m servers minimizing the number of servers used.

Examples

- **Server Consolidation problem**: The IT service manager needs to consolidate servers by assigning *n* services to target *m* servers minimizing the number of servers used.

$$minimize \sum_{\{i=1\}}^{m} c_i y_i$$

Equal to 1 if server i is used

$$s.t. \sum_{\{j=1\}}^{n} u_{jk} x_{ij} \leq s_{ik} y_i, \qquad \forall i \in I, \forall k \in \mathrm{K};$$

The total amount of resource requested from all services cannot exceed the capacity of the server for each resource

Each service must be deployed on exactly one server

$$\sum_{\{i=1\}}^{m} x_{ij} = 1, \qquad \forall j \in J;$$

$$y_i, x_{ij} \in \{0,1\}, \ \forall i \in I, \forall j \in \mathrm{J}.$$

Heuristics

Round Robin Fashion

1. Set of requests of VMs deployments
2. Number of servers $= \lceil \frac{max\_res\_VMs}{server_{cap}} \rceil$
3. VMs requests distributed in a Round Robin manner to the appropriate number of servers

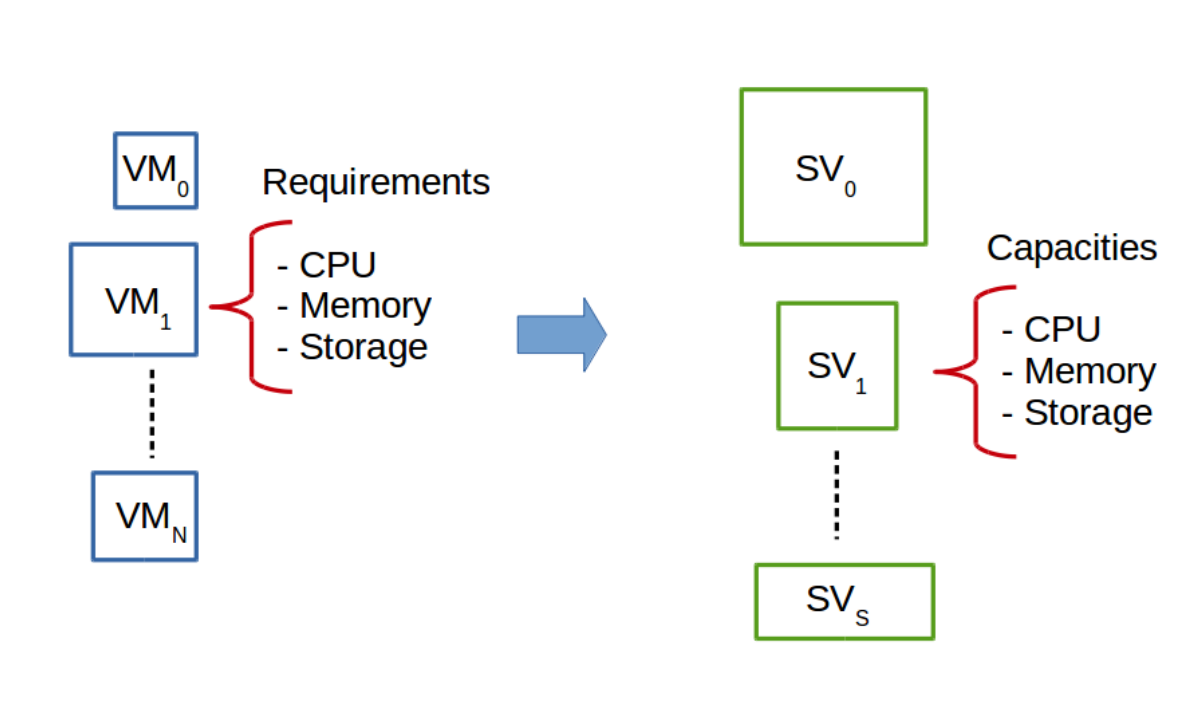
Heuristics

Round Robin Fashion

1. Set of requests of VMs deployments
2. Number of servers $= \lceil \frac{max\_res\_VMs}{server_{cap}} \rceil$
3. VMs requests distributed in a Round Robin manner to the appropriate number of servers

Drawbacks?

Multidimensional Case

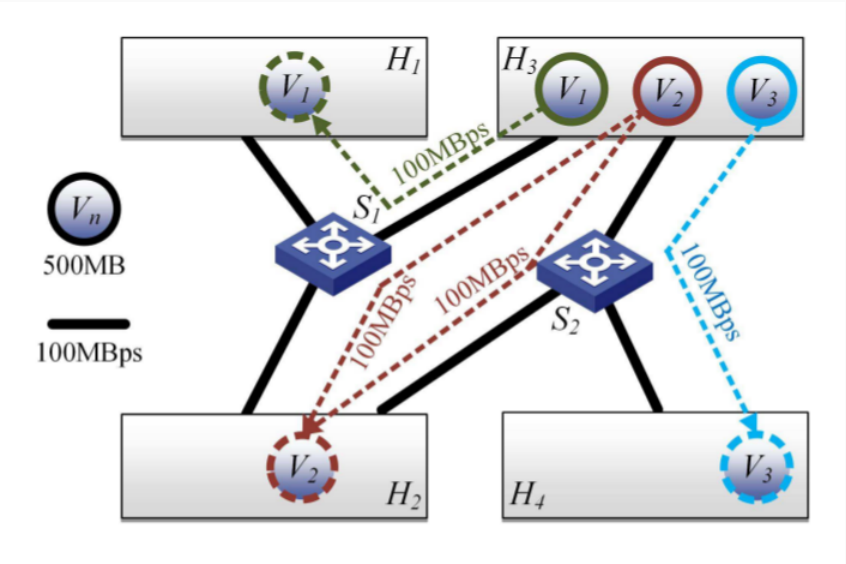


Dual Vector Bin-Packing

Dynamic Allocation

- Optimization approaches: not scalable
- Optimization phase: expensive in terms of time
- A controller can respond to overload or underload on a server during runtime and reallocates
- Possible through the service **live migration**

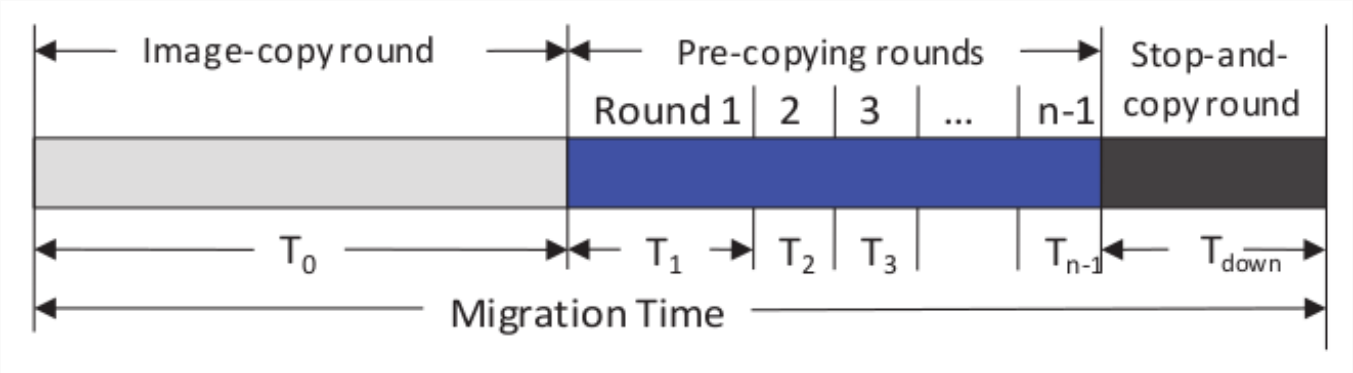
Live Migration



- "Moving VMs from one physical machine to another without disrupting services"[a]
- High availability
- Minimize the downtime
- Minimize the number of SLA violations
- Virtual machines: stateful

[a]H. Wang, Y. Li, Y. Zhang, D. Jin, "Virtual machine software-defined networks", IEEE Transactions on Clou

Service Migration Phases (VMs)



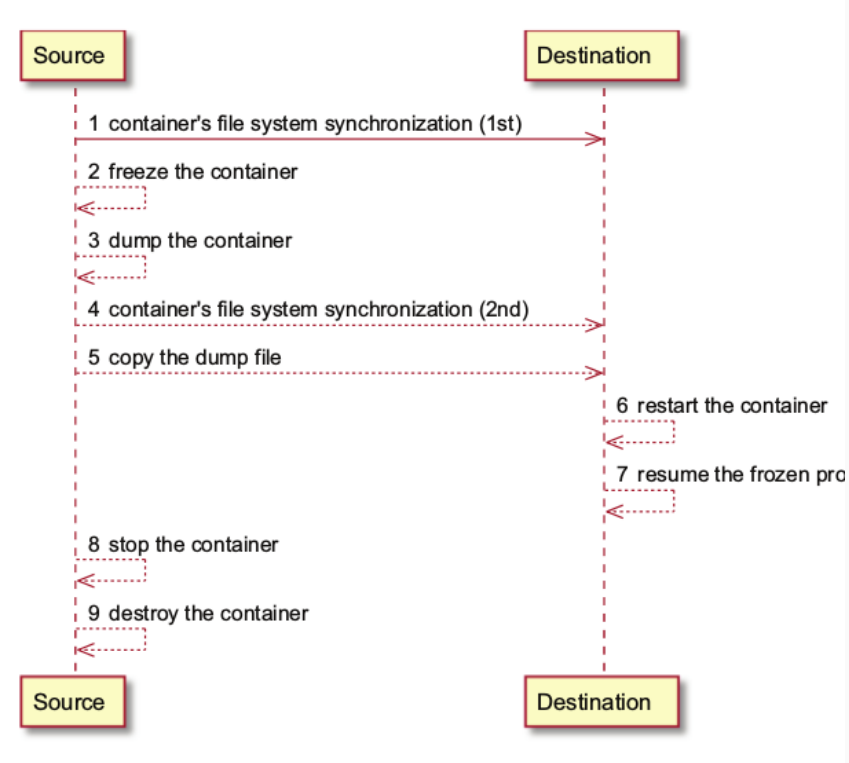
- Data volume transmitted at each round:

$$V_i = \begin{cases} M, & i = 0 \\ R \cdot T_{i-1}, & otherwise \end{cases}$$

- M: VM image
- R: copy rate

- Total migration time
- Downtime

Containers


Source
Destination
1 container's file system synchronization (1st)
2 freeze the container
3 dump the container
4 container's file system synchronization (2nd)
5 copy the dump file
6 restart the container
7 resume the frozen pro
8 stop the container
9 destroy the container
Source
Destination

Reactive Control

- A migration is triggered if the utilization of a server exceeds or falls below a certain threshold
- Monitoring system to receive the CPU and memory load of all servers and VMs in a three-second interval (typically)
- Data recorded and stored in a buffer for ten minutes
- Overload and underload situations detected by a control process running every five minutes
- **Thresholds**
- **Migration target servers**: chosen based on their volume

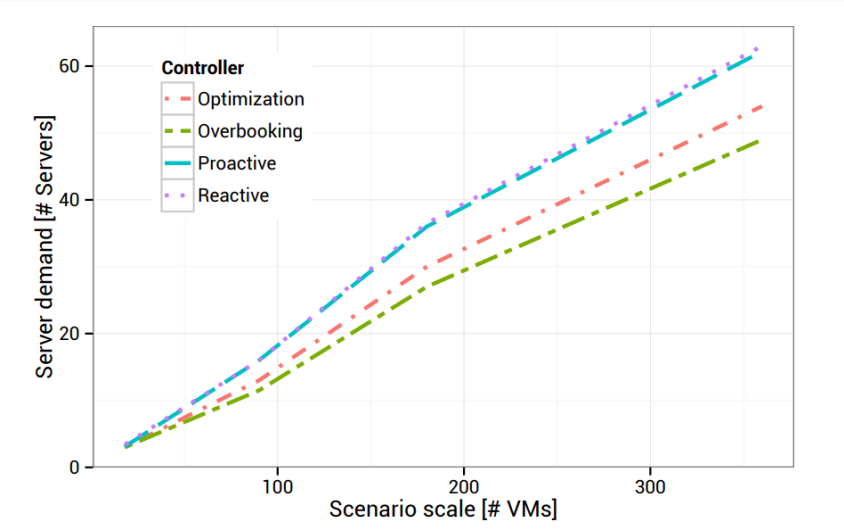
Proactive Control

- Extension of reactive control
- **Avoid unnecessary migrations**
- **Migration**: only triggered if the forecast suggests that the overload and underload continues and is not only driven by an unforseen spike in the demand
- For each server a load forecast is computed using one minute of utilization measurements
- If the forecast and M out of K measurements pass a threshold an overload or underload situation is detected
- **Forecast**: computed using *double exponential smoothing (DES)*

Static vs Dynamic

Dynamic allocation: often seen as the most efficient means to allocate resources in a data center. Is it always true?

Experimental results:



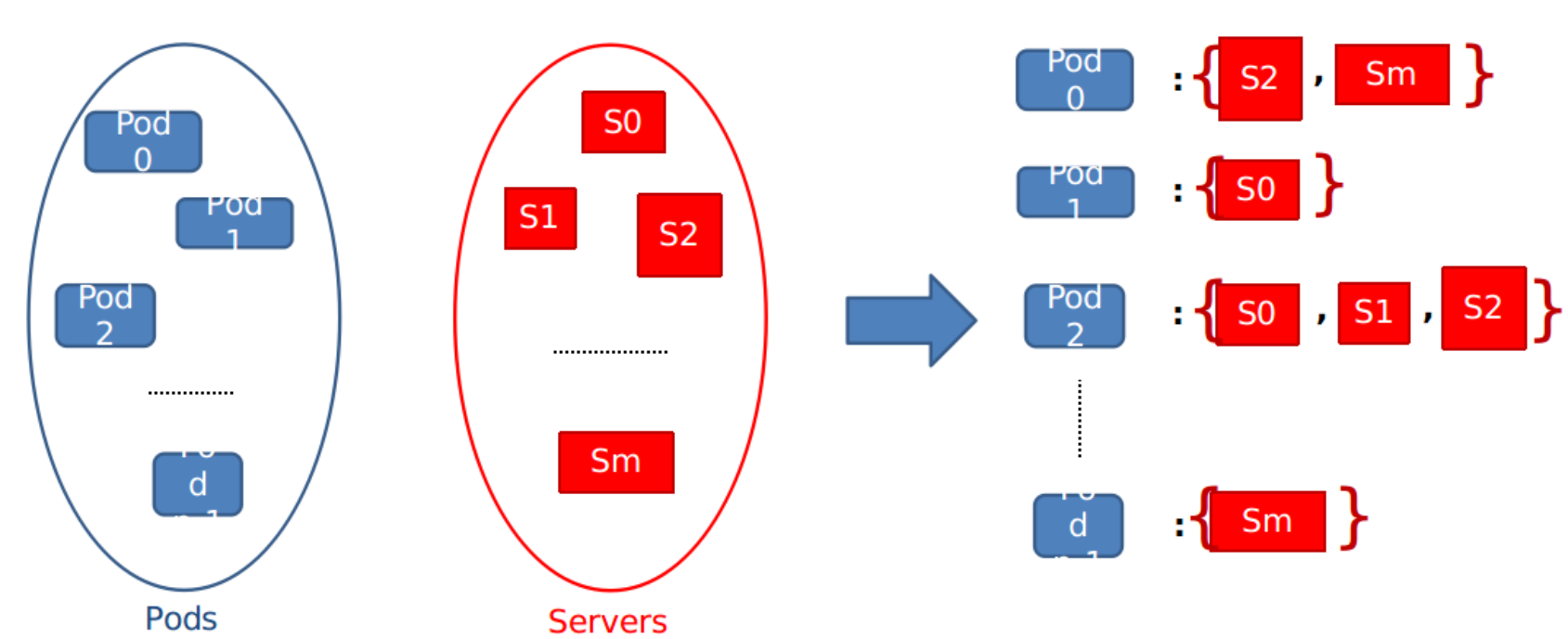
Suggestions: good combination of both mechanisms

- **Static allocation**: long term period allocation
- **Dynamic**: exceptional workload peaks

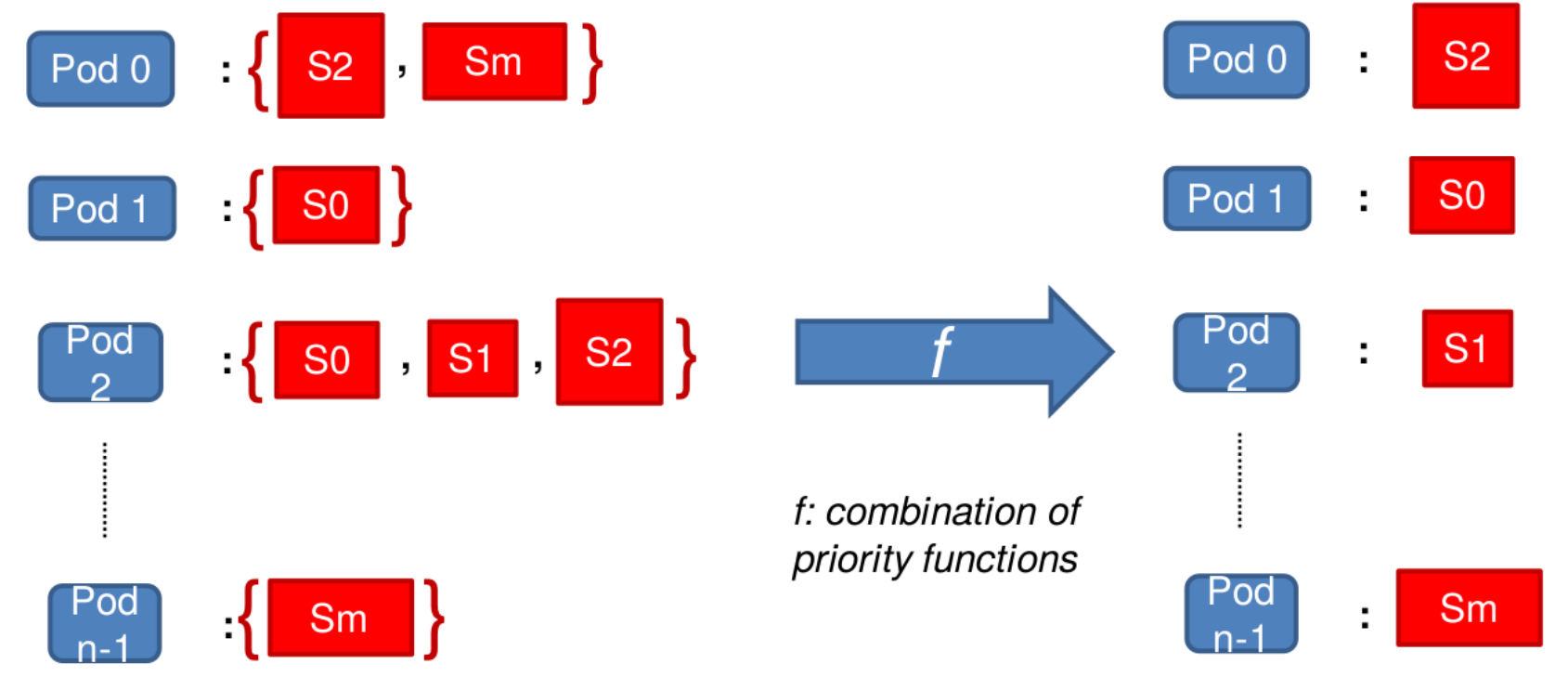
Well-known cases

- **Kubernetes**: "Open-source system for automatic deployment, scaling, and management of containerized applications"
- **Pod**: "Group of one or more containers with shared storage/network"
- **Heuristic solutions**:
  - **Input**: Set of servers of a cloud infrastructure; set of pods to be deployed;
  - Three main steps:
    1. *Filtering*
    2. *Ranking*
    3. *Deployment*

Filtering


Pod 0
Pod 1
Pod 2
Pods
S0
S1
S2
Sm
Servers
Pod 0 : { S2 , Sm }
Pod 1 : { S0 }
Pod 2 : { S0 , S1 , S2 }
Sm

Ranking


Pod 0 : { S2 , Sm }
Pod 1 : { S0 }
Pod 2 : { S0 , S1 , S2 }
Pod n-1 : { Sm }
f
f: combination of priority functions
Pod 0 : S2
Pod 1 : S0
Pod 2 : S1
Pod n-1 : Sm

Priority Functions

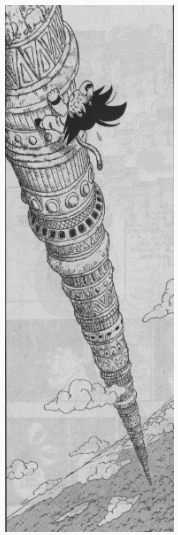
- **LeastRequestPriority**: prefers the node with highest free fraction (in terms of CPU and memory)
- **CalculateNodeLabelPriority**: prefers nodes with specified labels
- **BalancedResourceAllocation**: CPU and Memory balanced after the Pod deployment
- **CalculateSpreadPriority**: minimizes the number of pods belonging to the same service on the same node
- **CalculateAntiAffinityPriority**: minimizes the number of pods of the same service on nodes with the same value for a particular label

Resource allocation in the Fog

Problem (1)

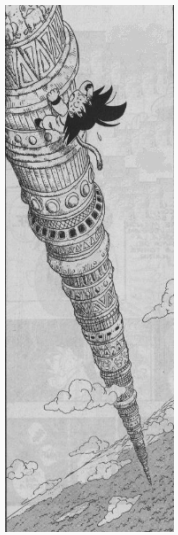
Problem (1)

- Cloud: far (high latency)



Problem (1)

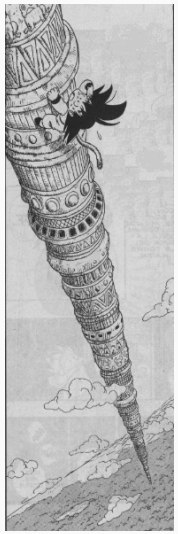
- Cloud: far (high latency)



- High costs

Problem (1)

- Cloud: far (high latency)

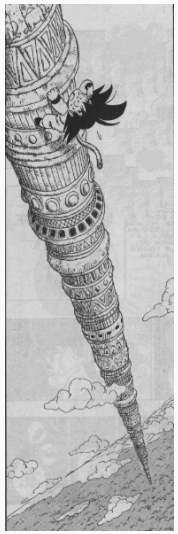


- High costs

- Privacy issues

Problem (1)

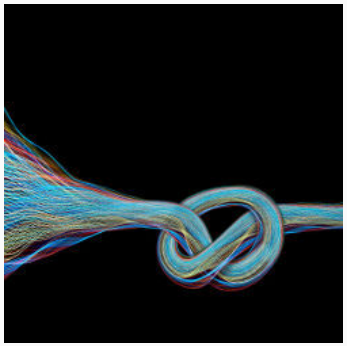
- Cloud: far (high latency)



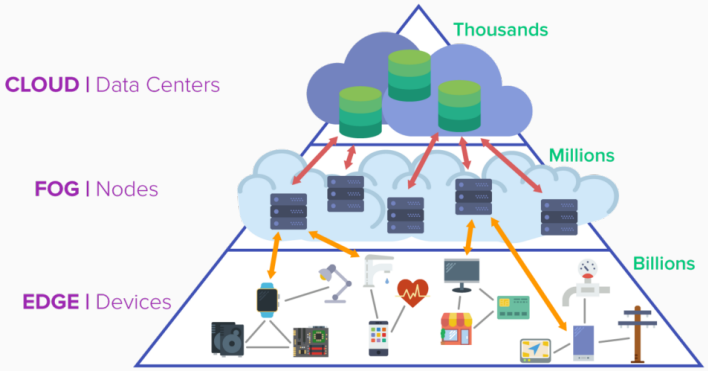
- High costs

- Privacy issues

- Network congestions

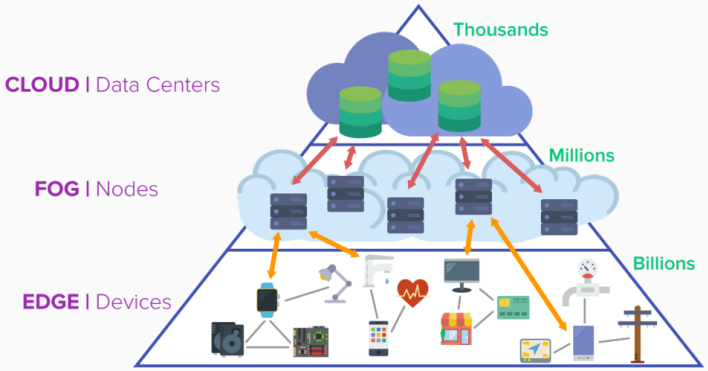


Fog Computing



- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution

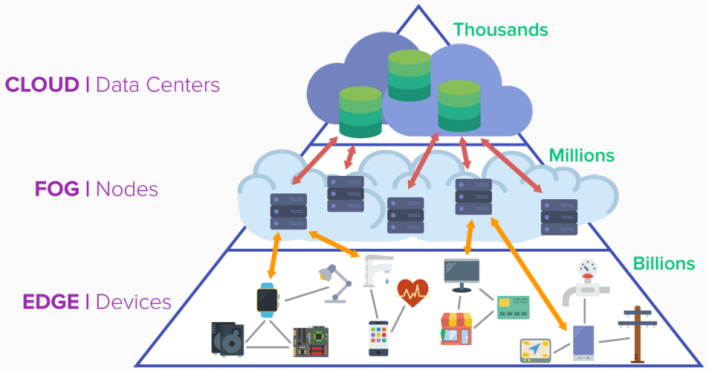
Fog Computing



- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution

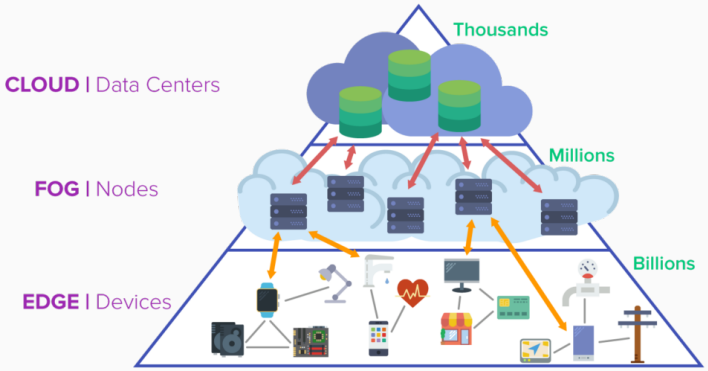
- Reduction of data traffic towards the cloud

Fog Computing



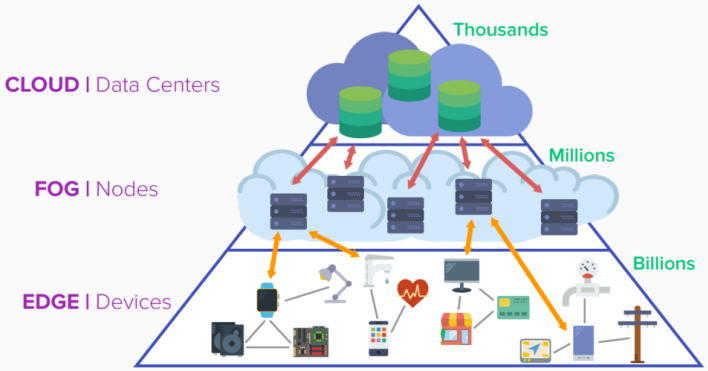
- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution
- Reduction of data traffic towards the cloud
- Low latency

Fog Computing



- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution
- Reduction of data traffic towards the cloud
- Low latency
- Support for mobility

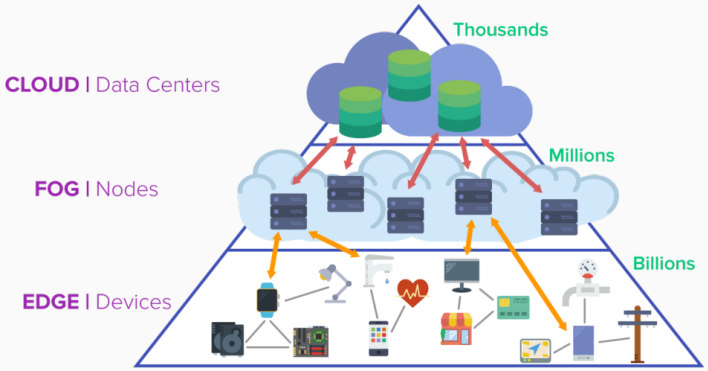
Fog Computing



- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution

- Reduction of data traffic towards the cloud
- Low latency
- Support for mobility
- Reduced costs

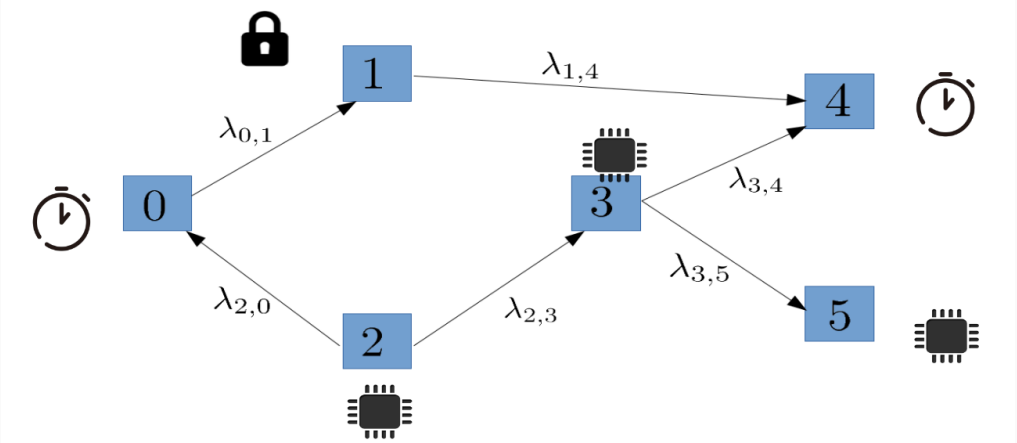
Fog Computing



- Extension of cloud computing
- Fog nodes: access points, routers, switches, base stations, servers
- Computation closer to things
- Geographical distribution

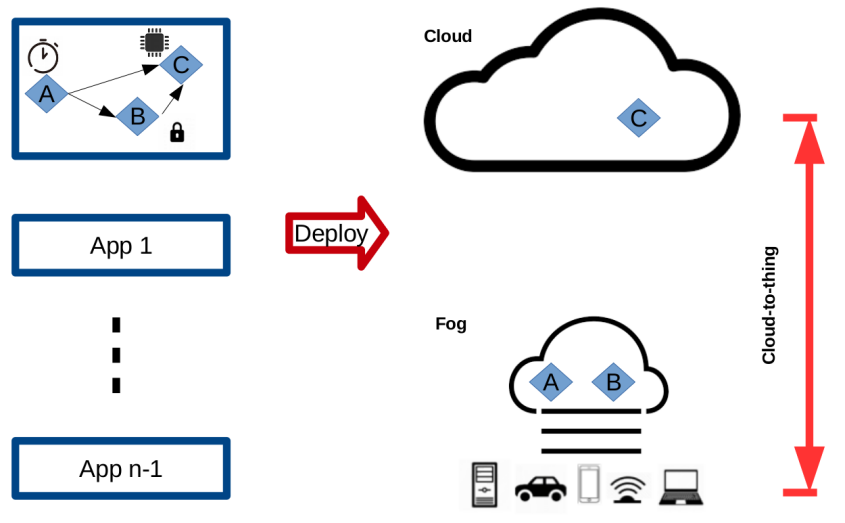
- Reduction of data traffic towards the cloud
- Low latency
- Support for mobility
- Reduced costs
- Privacy improved

IoT Applications



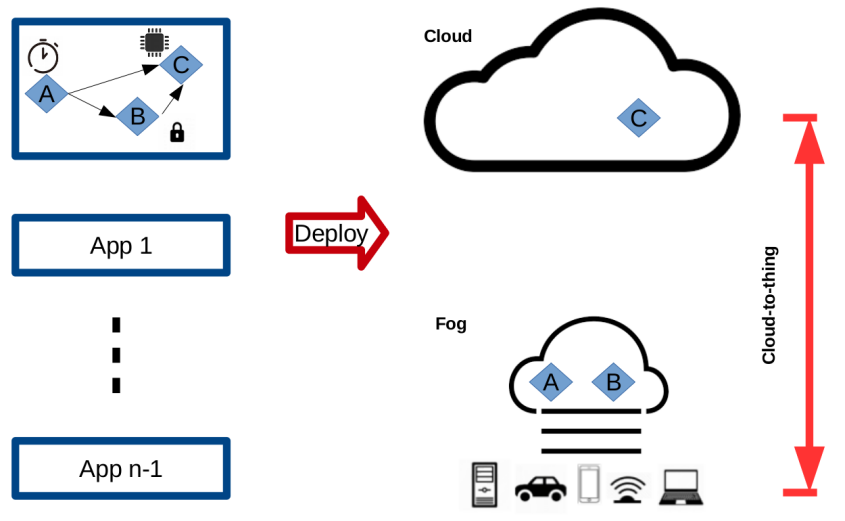
- Microservice architecture
- Multiple coupled modules
- Different requirements (CPU, memory, storage and bandwidth)

Challenge

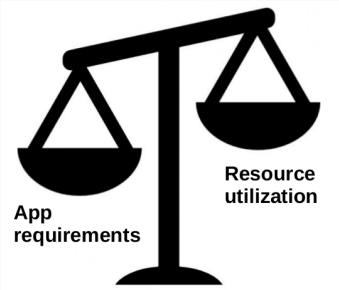


- Application modules: displaced on the cloud-to-things continuum
- Heterogeneous nodes

Challenge



- Application modules: displaced on the cloud-to-things continuum
- Heterogeneous nodes



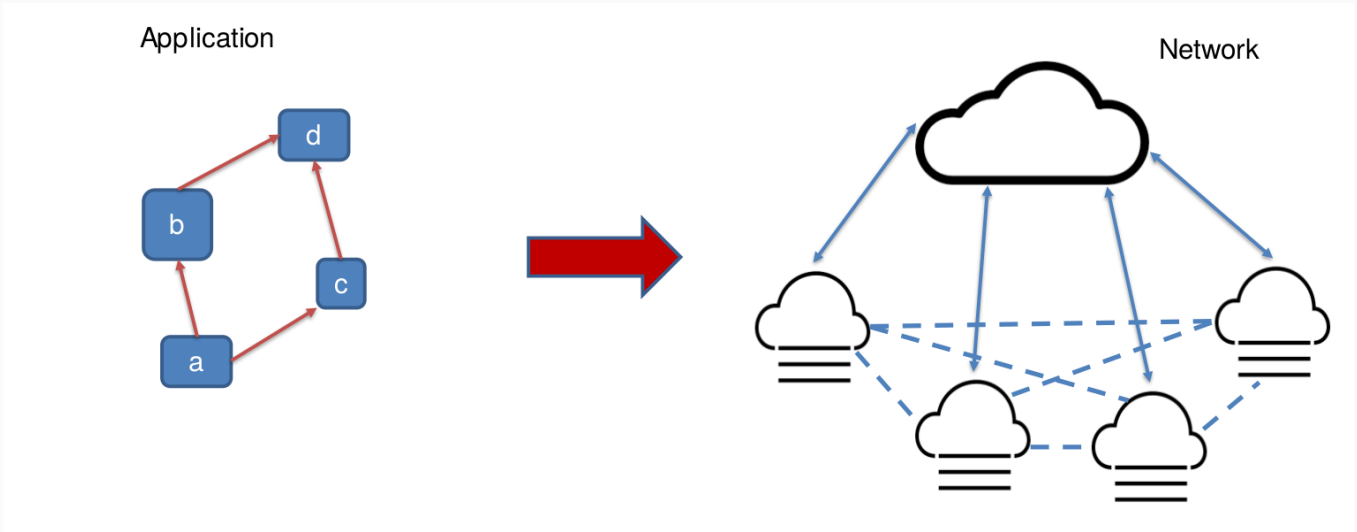
Resource Allocation in Fog

- Fundamental role in the trade-off between resource utilization and application requirements satisfaction
- **Two approaches**:
  - Static
  - Dynamic
- **Two players**
  - Infrastructure owner
    - *Objective*: maximize the revenue, maximizing the number of applications deployed on the infrastructure
  - Application owner
    - *Objective*: minimize the deployment cost and the completion time

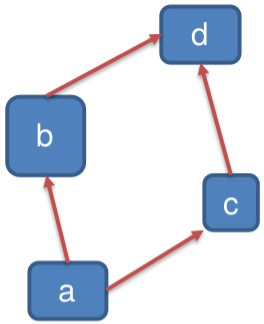
Static Allocation

- Applications and Network graphs
- Application deployment reduces to Virtual Network Embedding (VNE) problem (variant of Subgraph Isomorphism): NP-hard
- Approximation algorithms
- Heuristics:
  - Node ranking for each application module
  - Routing between deployed modules

Static Allocation

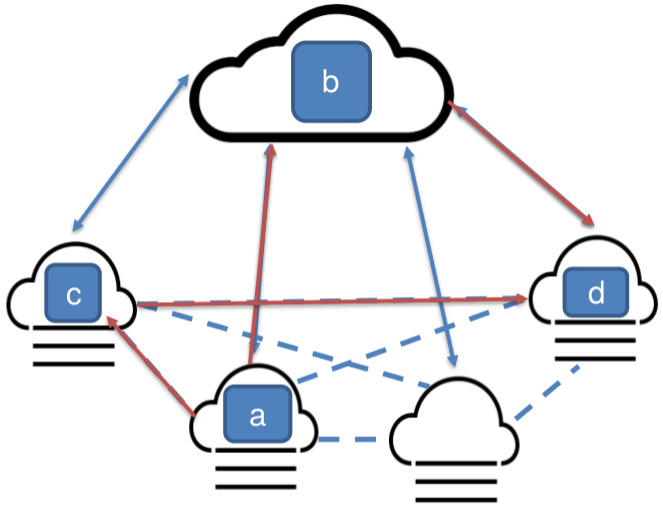


Static Allocation



Application sorting:

- {a,b,c,d}
- {a,c,b,d}
- {a,b,d,c}
- {a,c,d,b}



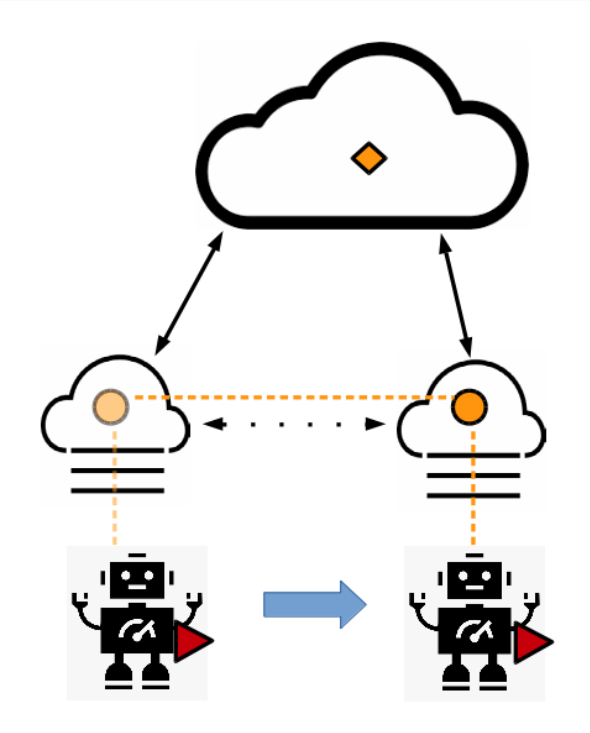
1. Placement (Node Mapping)
2. Routing (Link Mapping)

Dynamic Allocation

- Different motivations w.r.t. the Cloud
- Live migration is applied to:
  - Deal with the geographical distribution
  - Save money on the cloud
  - Better utilize fog resources
  - Guarantee a good level of QoS

Dynamic Allocation

Industrial scenario



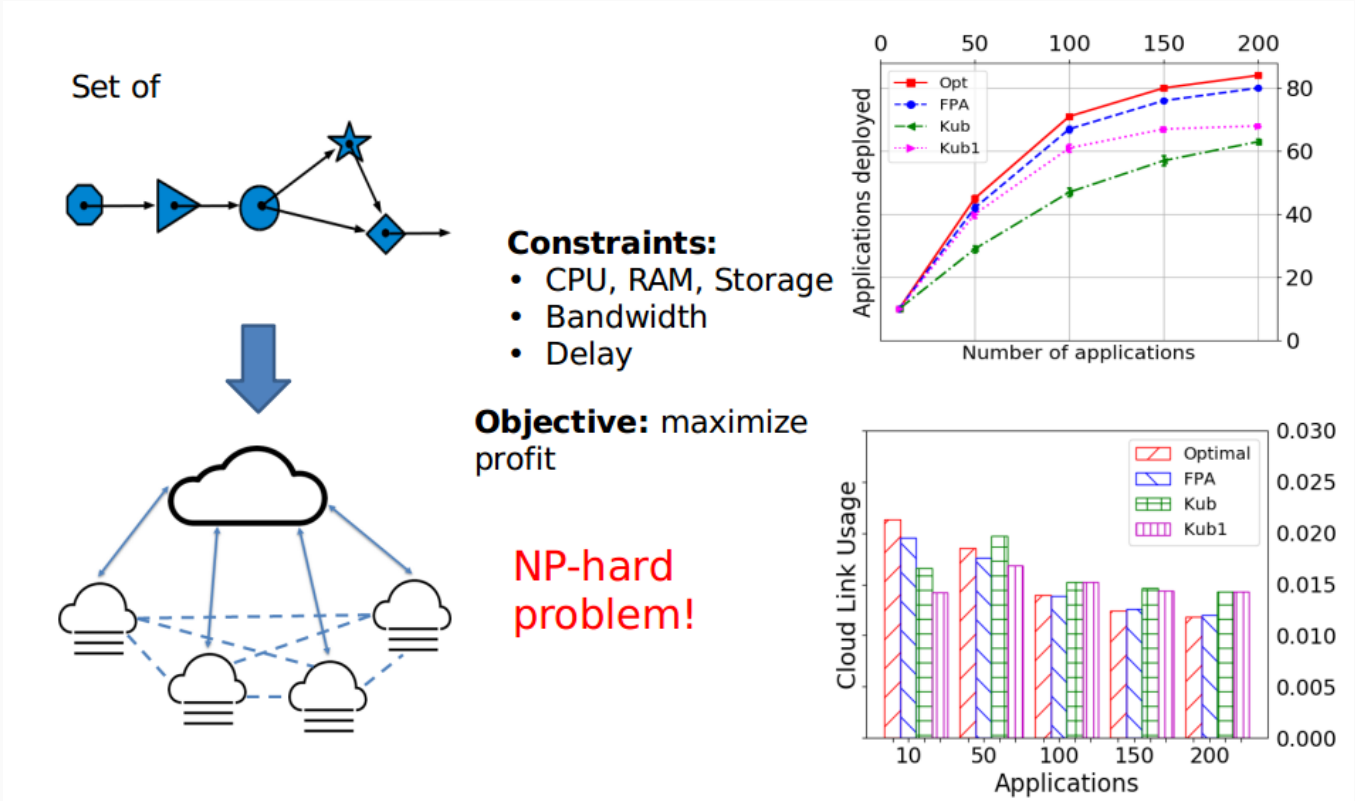
- Monitoring the nodes occupation
- Monitoring clouds costs
- Verifying the resource availability of target nodes
- Perform the migration

Static vs Dynamic in Fog

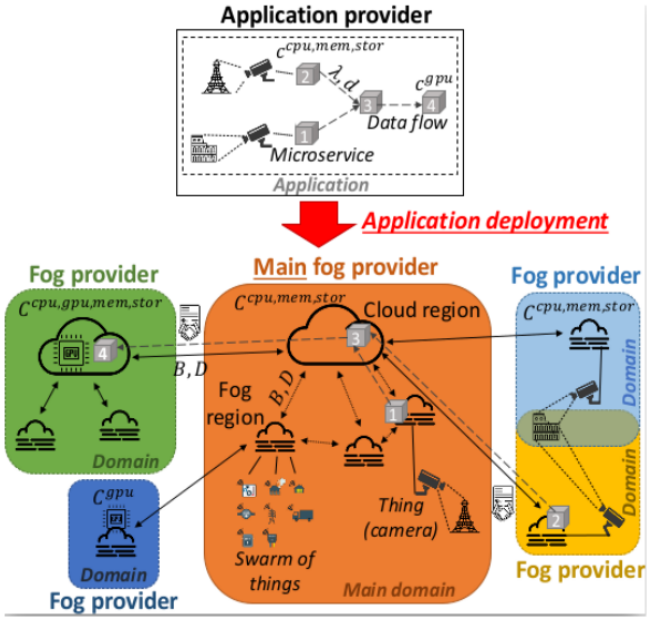
- Complete knowledge of the application requirements and network availability
- Optimization problem
- NP-hard problems
- Approximated solutions

- Partial knowledge
- Variability of applications performances
- Live migration
- Monitoring systems
- Application scaling

Research (Static Allocation)



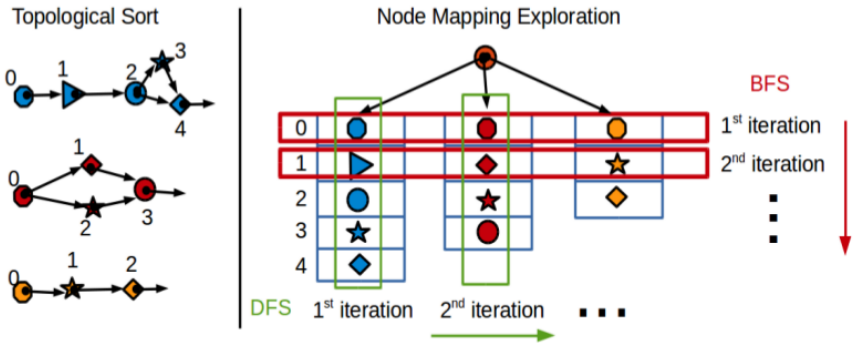
Federated Fog



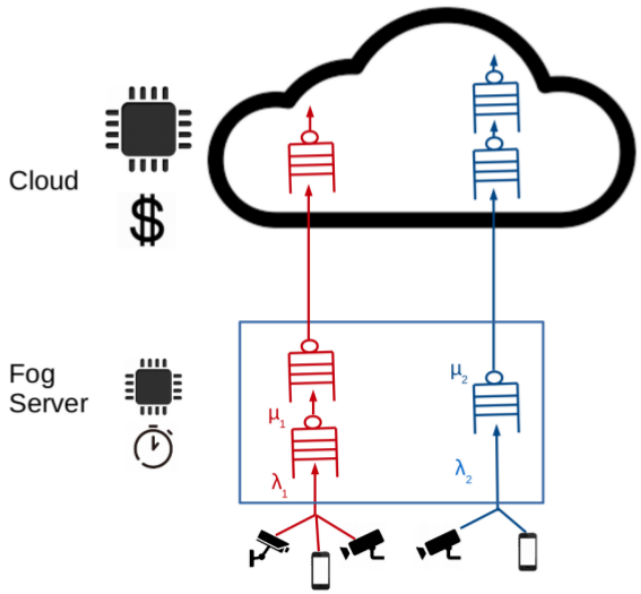
Constraints:

- CPU, RAM, Storage
- Bandwidth
- Delay
- Locality constraints

Objective: minimize cost



Dynamic Allocation

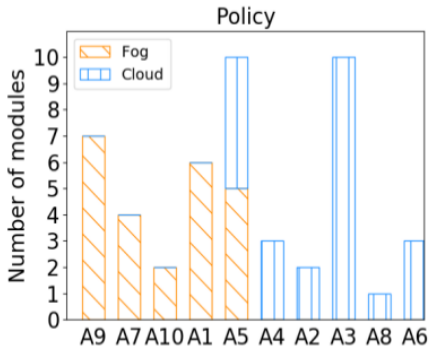


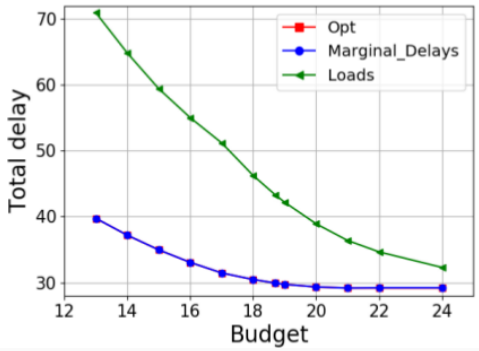
Processor Sharing

Constraints:
- Budget
- Stability

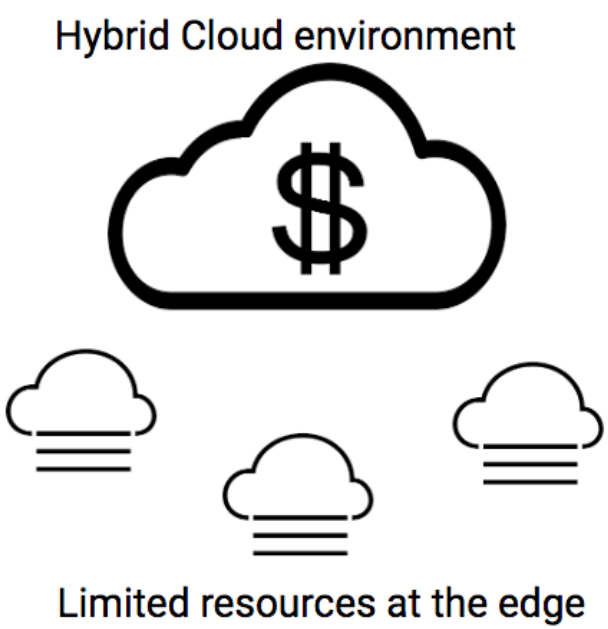
Objective: minimize total average delay

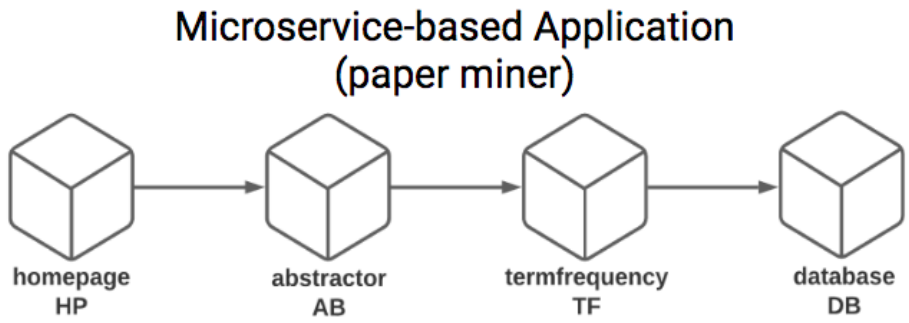
Marginal delays: increase of processing delay in Fog





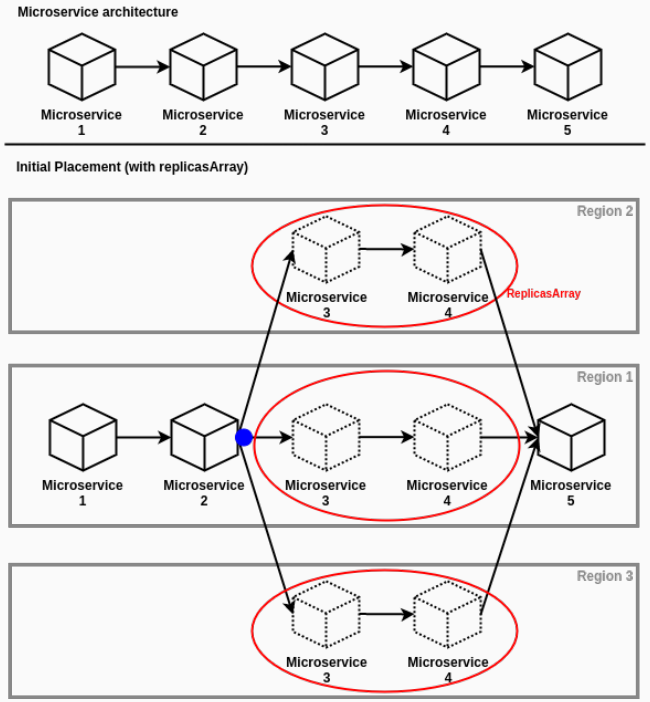
Hybrid Scenario

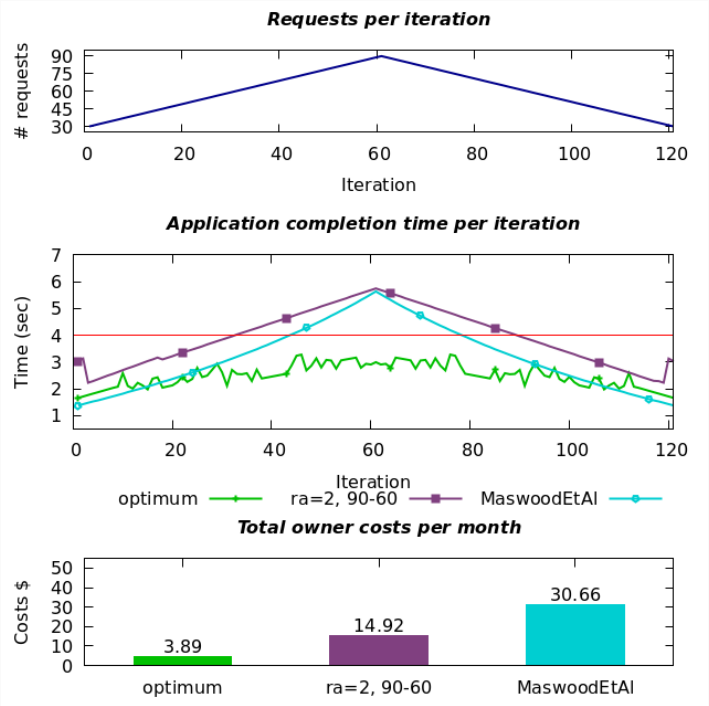




Objective: Minimize public cloud cost while guaranteeing application's QoS requirements

Cost-Effective Workload Allocation Strategy





- Orchestration problem with replicas
- NP-hard problem

- Trade-off: Cost vs. Completion Time

Take-home Message

- Resource allocation: "old-fashioned" problem
- Strong theoretical foundations
- Fundamental role in distributed systems
- Different problems depending on the context
- Fog and Cloud: new challenges and objectives
- New algorithmic solutions

Material

Readings:

- A. Wolke, Ma. Bichler, and T. Setzer «Planning vs. Dynamic Control: Resource Allocation in corporate Clouds», IEEE Transactions on Cloud Computing, 2016
- B. Speitkamp and M. Bichler, «A mathematical programming approach for server consolidation problems in virtualized data centers», IEEE Trans. Services Comput., 2010
- S. Akoush, R. Sohan, A. Rice, A.W. Moore, and A. Hopper, «Predicting the performance of virtual machine migration», in Proc. Int. Symp. Model.,Analys.,Simul.,Comput. Syst,, 2010

Additional readings:

- M. Sipser, «Introduction to the Theory of Computation» (Book)
- F. S. Hillier, G. J. Lieberman, «Introduction to Operations Research» (Book)
- T. Ibaraki and N. Katoh, «Resource Allocation Problems: Algorithmic Approaches» (Book)

Quote

“Nebulat ergo cogito”