Resource management in distributed systems

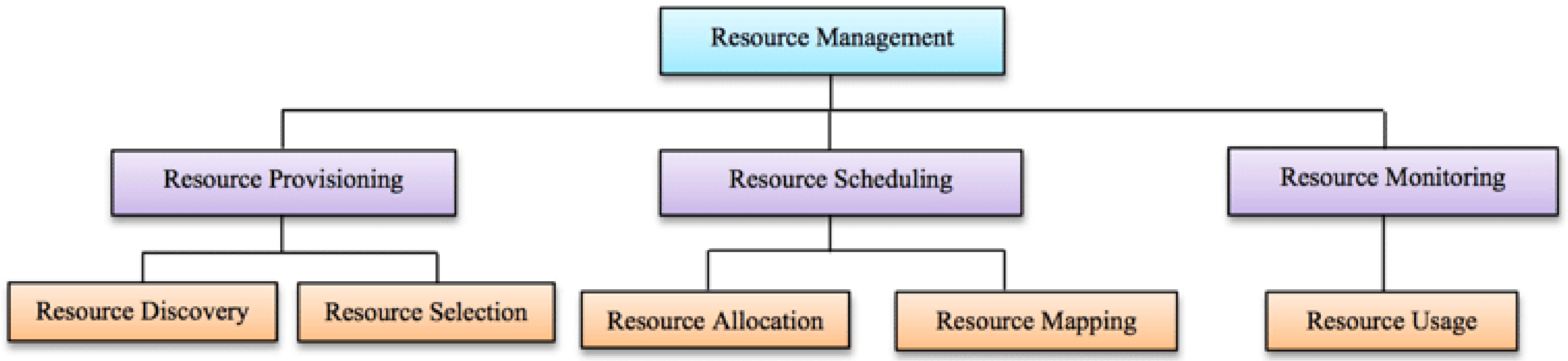
Resource management

Topic in various fields such as:

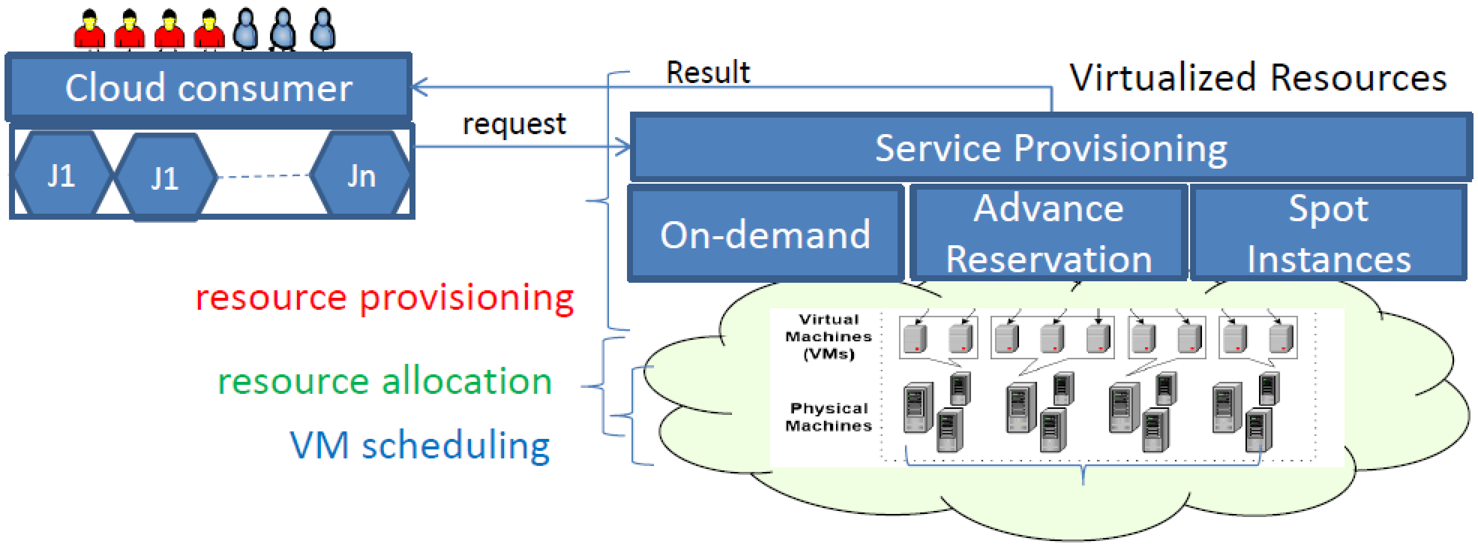
–job shop, flow shop or open shop scheduling in production environment, printed circuit board assembly scheduling

–scheduling of tasks in distributed computing systems such as cluster, grid, cloud, edge, fog

Basic taxonomy in this lecture



Example for IaaS (Cloud)



Resources in IaaS (Cloud)

Provisioning

Is responsible

–To understand the user needs

–To prepare VMs with appropriate resources to match the workloads and QoS requirements

–SLA Negotiation

–Etc.

Allocation

Is responsible

–To select an optimal set of physical machines to host the received services (VMs),

–To ensure the resource and QoS constraints are met.

–To manage changes in resources availability through VMs restore or migration

Goals in provisioning

- A wide variety of resource provisioning goals exist:
  - –High resource utilization
  - –Energy efficiency
  - –Reliability of services
  - –Low performance interference
- Optimally achieving the goals in cloud computing environment has been proved to be an NP-complete problem due to its combinatorial optimization nature.
  - There are no algorithms which may produce optimal solution within polynomial time for such kind of problems.

Inputs for provisioning algorithms

- Is imperative to consider information such as highly heterogeneous and time-varying workloads.
  - –daily demand distribution of a typical Internet application
  - –request arrivals and departures statistics
- What type of resources the application needs?
  - –VM that is abundant in a particular type of resource: bandwidth, CPU, or storage.
- aspects such as
  - –the multi-tenant,
  - –resource-abundant,
  - –elastic resource model and
  - –quality of service requirements expressed in terms of execution time and cost

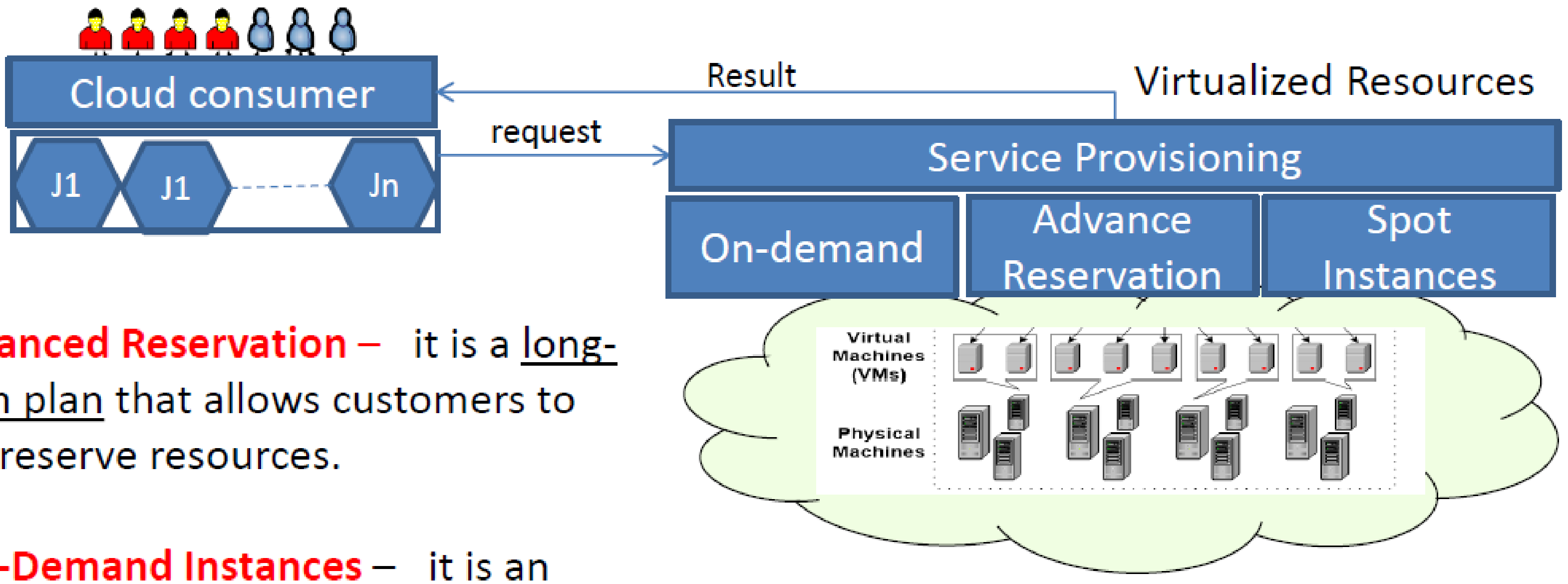
Resource provisioning classification

- Static Resource Provisioning
  - Given information such as highly heterogeneous and time-varying workloads , allocation solutions based on static resource provisioning lead to poor performance and hinder providers from achieving expected profits.
- Dynamic Resource Provisioning
  - With dynamic provisioning, the provider allocates more resources as they are needed and removes them when they are not.
- Hybrid Resource Provisioning
  - Combines other resource provisioning approaches

Provisioning in IaaS (Cloud)

- The Cloud resource provisioning enables virtualized resources to be allocated to Cloud consumers based on three provisioning plans
  - –On-Demand,
  - –Reserved, and
  - –Spot instances (Preemptible VMs in Google)

The three provisioning plans



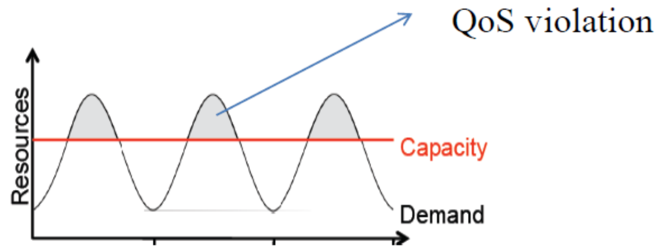
Advanced Reservation – it is a long-term plan that allows customers to pre-reserve resources.

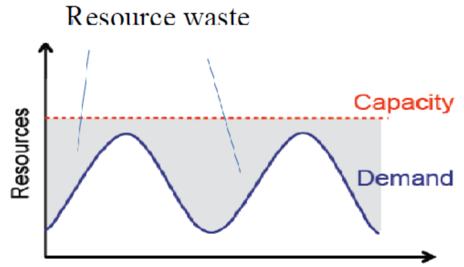
On-Demand Instances – it is an intermediate-term plan that allows customers to pay hourly for resource usage on pay-as-you-go bases.

Spot Instances - it is a short-term plan that allows customers to bid on unused resources.

Under vs over-provisioning

- Planning resources for only usual workloads request rejection and QoS degradation





Over-provisioning to handle demand peaks can result in significant costs and unused capacities

Under-provisioning or over-provisioning?

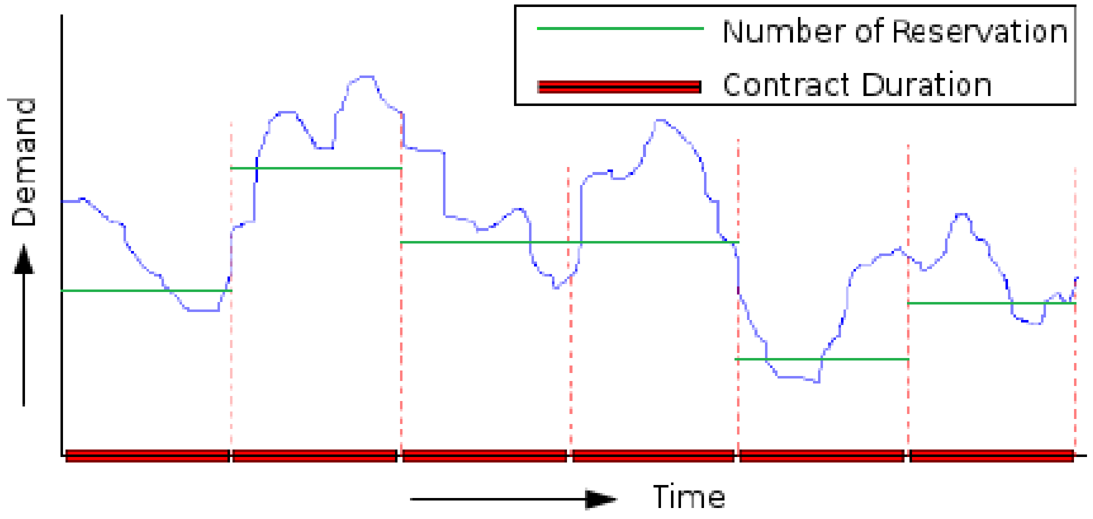
- Minimizing both under-provisioning and overprovisioning problems under the demand and price uncertainty in cloud computing environments is important.
- The under-provisioning problem can be solved by provisioning more resources at higher cost with on-demand plan.
- The only recourse for the overprovisioning problem is to support it with on-demand instances fee structure

Advanced reservation

- Some applications such as interactive multimedia require end-to-end reservation for resources.
- Advance reservation techniques are very useful in federated cloud as well, particularly for the co-allocation of various resources
- Reservation is accepted for a fixed contract duration (e.g. for a 1 year contract or for a 3 year contract) with a one-time upfront payment for the duration of the contract
- Reservation is significantly cheaper than the on-demand fee structure.

Single Contract reservation

- The reservation decision is taken for each contract duration (called segment) separately during the whole duration of the demand vector as shown below



Spot Instances

- Spot instances are Amazon’s third plan specifically tailored for offering their unused resources at a much lower cost than both on-demand and advanced reservation
- Major cloud service providers (AWS, Google, and Azure) offer the option to use Spot Instances.
- Cloud users can bid on unused Amazon EC2 capacity and run those instances for as long as their bid exceeds the current spot price.
- A wide variety of auction-based approaches for spot instances bidding has been proposed in the literature

When to use spot instances?

- Very useful in both batch processing and high-performance clusters, as well as web server fleets with variable workloads.
- Well-suited for data analysis, batch jobs, background processing, and optional tasks
- Spot Instances are a cost-effective choice if you can be flexible about when your applications run and if your applications can be interrupted.
- The appeal for Spot Instance is its cheapness as compared to both Reserved and on-demand Instances
- It is estimated that

–reserved instances can save up to 70% compared to On-Demand with a 1- or 3-year commitment

–spot instances saving is as high as 80-90% compared to On-Demand

Challenges

- The problem with spot instances is that their price changes periodically based on supply and demand of spot instances
- Cloud users whose bid exceeds the current spot instances price gain access to the available spot instances.
- This render the use of spot instances unreliable since the instances may become unavailable at any time without any notice to the customer.
- One way to handle the sudden discontinuation of spot instances is to deploy checkpointing schemes

Spot Instances vs. On-demand instances

–Spot Instances can only be launched immediately if there is available capacity,

–the hourly price for Spot Instances varies based on demand, and

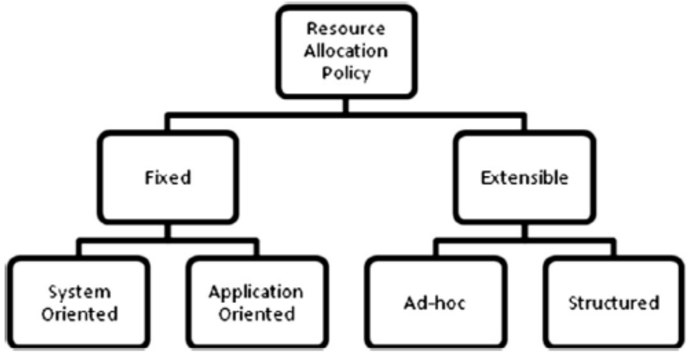
–Spot instances can be interrupted

Resource allocation: (1) in Grids

- Resource allocation of requests= „ Matchmaking”
- Globus Architecture for Reservation and Allocation (GARA)
  - Characterized by a *Start Time* and an *End Time*
  - Guaranteed service – QoS assurances
- Matchmaking “in the dark”
  - Advance Reservation (AR) Request:
    - Earliest Start Time
    - Execution Time
    - Deadline
  - Focus: Matchmaking without knowing the resource scheduling policy

Grid: resource allocation policy

- Fixed
  - Application oriented policy: GRACE, Ninf, G-QoSM, Javelin, 2K, AppLeS, Cactus, PUNCH, Nimrod/G, NetSolve
  - System oriented: Darwin
- Extensible
  - ad-hoc: MOL, **Globus**
  - structured: Legion



Resource allocation: (2) in Clusters

- Centralized: LoadLeveler, LSF, PBS, SLURM, Condor, PVM, Libra, OpenSSI, Faucets, GNQS
- Decentralized: Cluster-on-demand, Kerrighed, Gluster, DQS, Tycoon, Enhanced MOSIX, REXEC

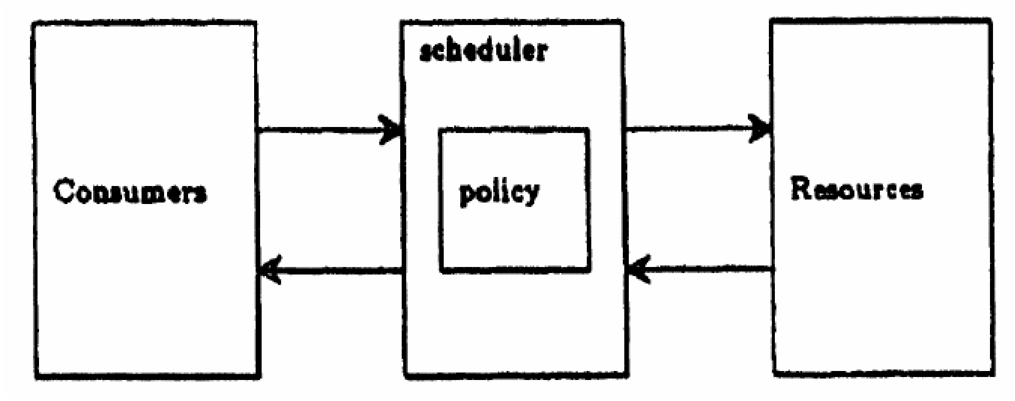
Resource allocation: (3) in Clouds

- **Resource discovery** identifies the physical hosts that are available that best suites the user requirements
- **Resource scheduling** aims to identify the best resource from a set of available matched resources.
  - **Resource allocation is performed that aims to allocate the selected resource to the job**.
    - The process in its broader terms is carried out by a broker who selects the best VM instance against set of available VM instances for the purpose of meeting QoS requirements of each product operation.
- Resource allocation is followed by **resource monitoring**

Resource scheduling: the problem

"how to distribute (or schedule) the processes among processing elements to achieve some performance goal(s), such as minimizing execution time, minimizing communication delays, and/or maximizing resource utilization"

- **A restatement of the classical problem of job sequencing in production management**
- **A scheduling problem consists of three (logical) components:**



The scheduling problem definition

Let $\mathcal{W}$ and $\mathcal{R}$ represent the static aspects of the workload and resources respectively

Let $\mathcal{Q}$ be the set of scheduling requirements that must be satisfied, including the scheduling goal pursued.

We define a scheduling problem $\mathcal{SP}$ as a tuple $(\mathcal{W}, \mathcal{R}, \mathcal{Q})$.

A scheduling problem may be stated as the following optimization problem.

Optimize	$\mathcal{Q}$.goal
Subject to	
	Static features of $\mathcal{W}$
	Static features of $\mathcal{R}$
	Requirements in $\mathcal{Q}$

A scheduling solution $\mathcal{SS}$ must always be associated with a given scheduling problem.

Even simple scheduling problem is complex!

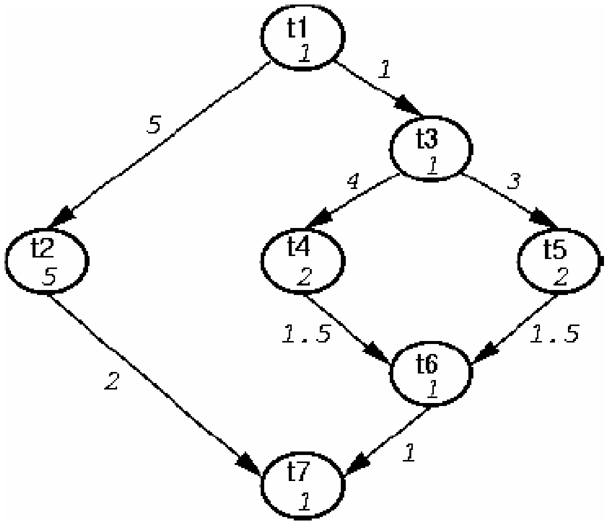
- Consider this simple scenario:
  - 2 processors, same capacity
  - N processes, different execution times, known a-priori (static scheduling)
  - No dependencies, no communication cost…
  - GOAL: minimize completion time of the processes set.
- If a schedule exists in which the processing loads are equal with no unnecessary delay, this is optimal!
- Finding such a schedule maps straightforwardly to the NP-complete "set-partitioning" problem:
  - Given a set of integers A and an integer size s(a), find A' subset of A s.t.:

$$\sum_{a\in A'} s(a) = \sum_{b\in(A-A')} s(b).$$

Scheduling can be more complex ….

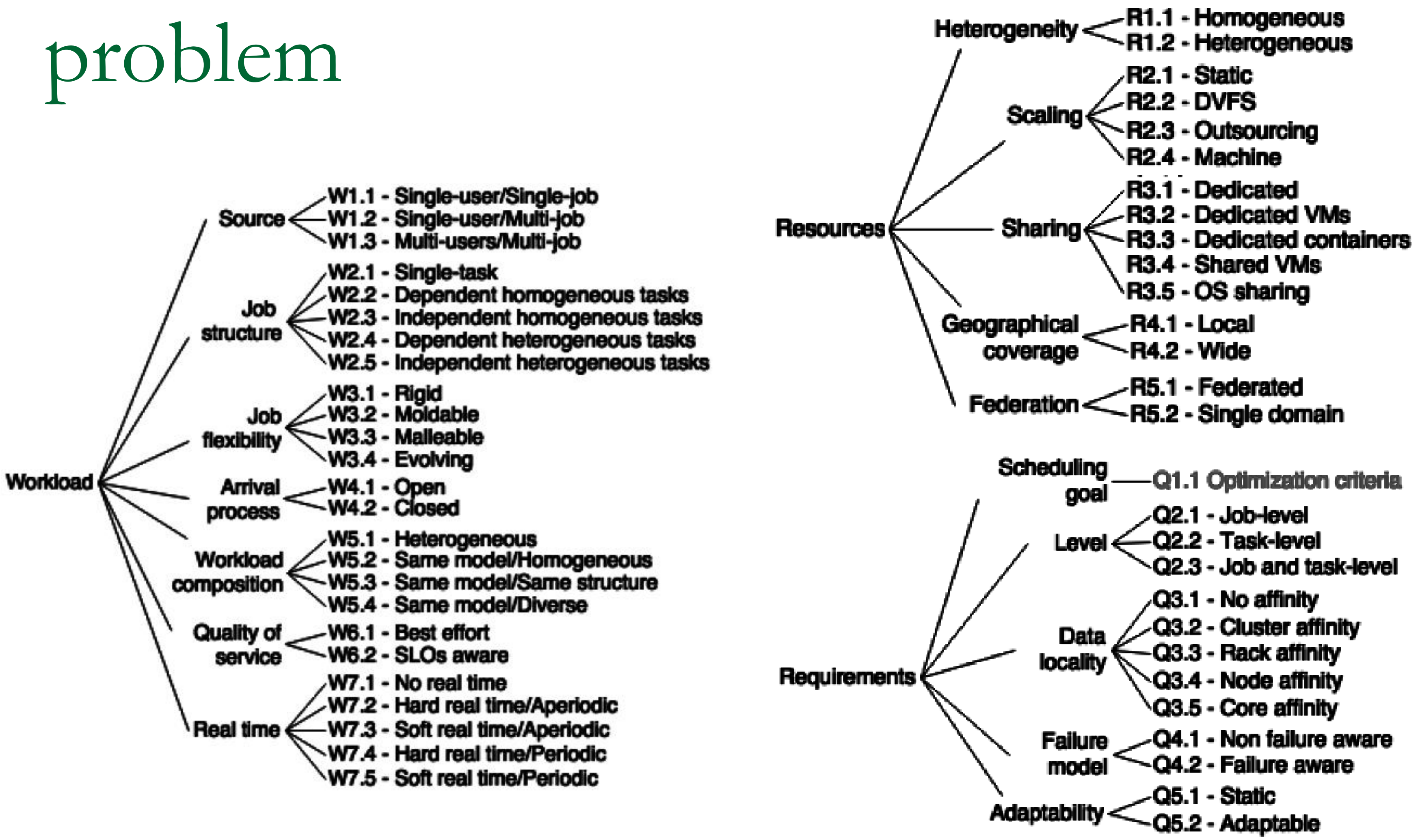
Essentially with additional constraints, e.g. :

- Task dependencies:
  - Example: parallel programming
  - Modelled by Direct Acyclic Graphs (DAGs)
    - Vertex is a task, its weight representing the computational cost
    - Arc expresses causal dependency, its weight representing the communication cost

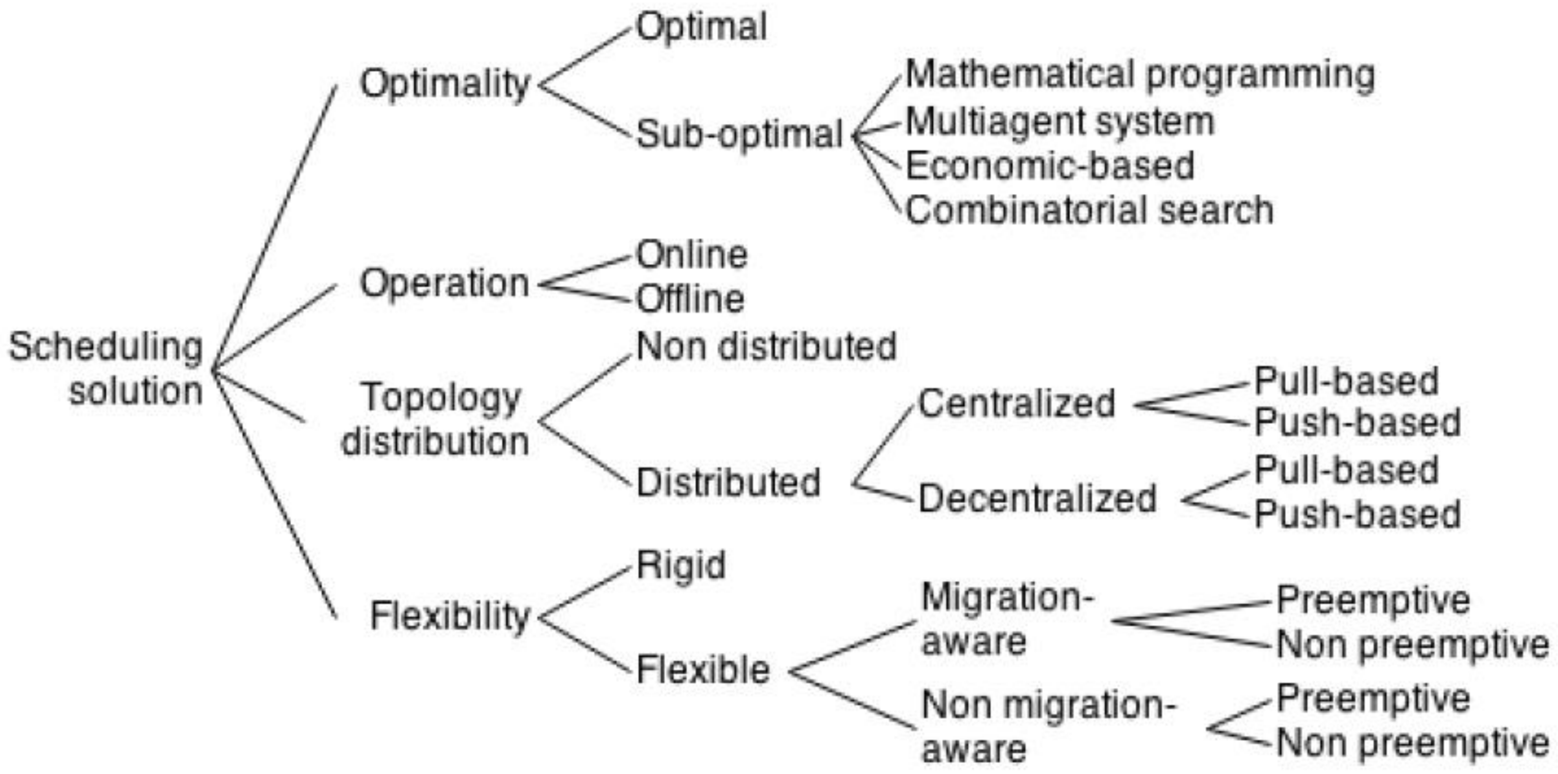


- Meeting task deadlines:
  - Example: Real Time Systems
  - In this case a feasible scheduling may even not exist!

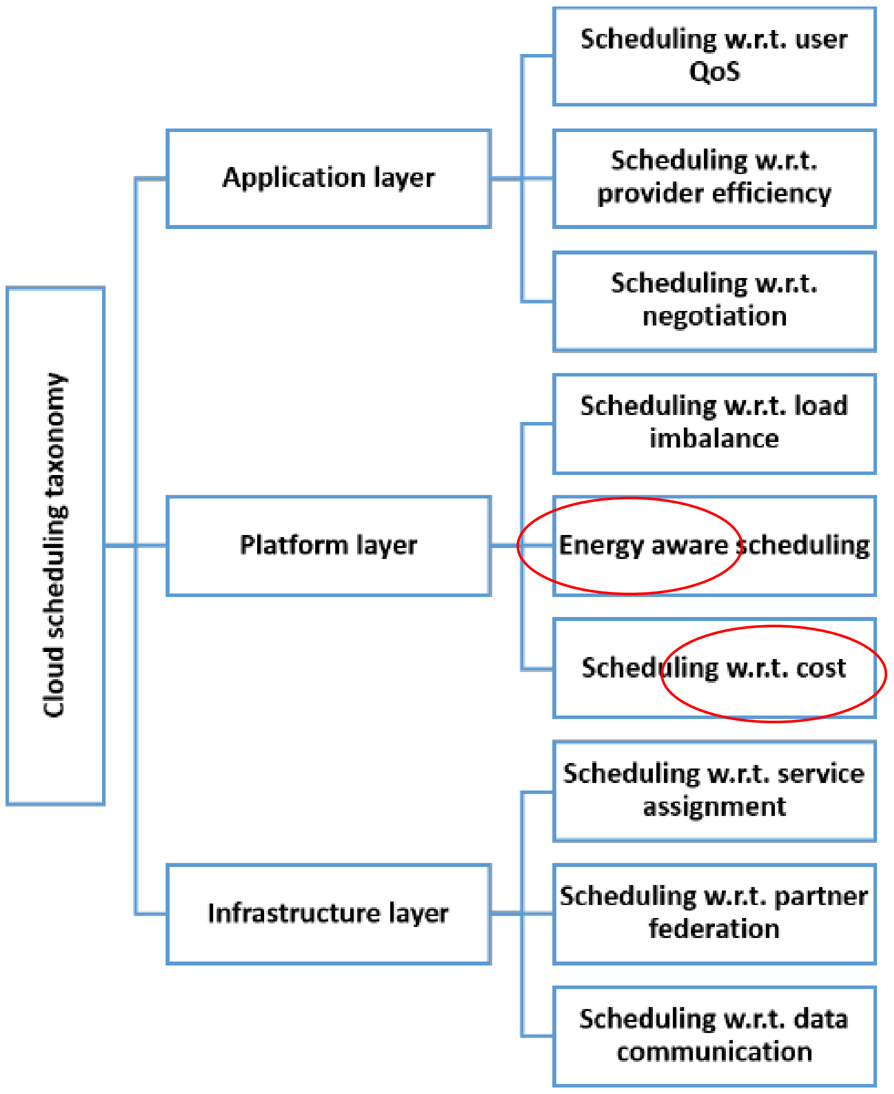
Static features related to a scheduling problem



Scheduling solution



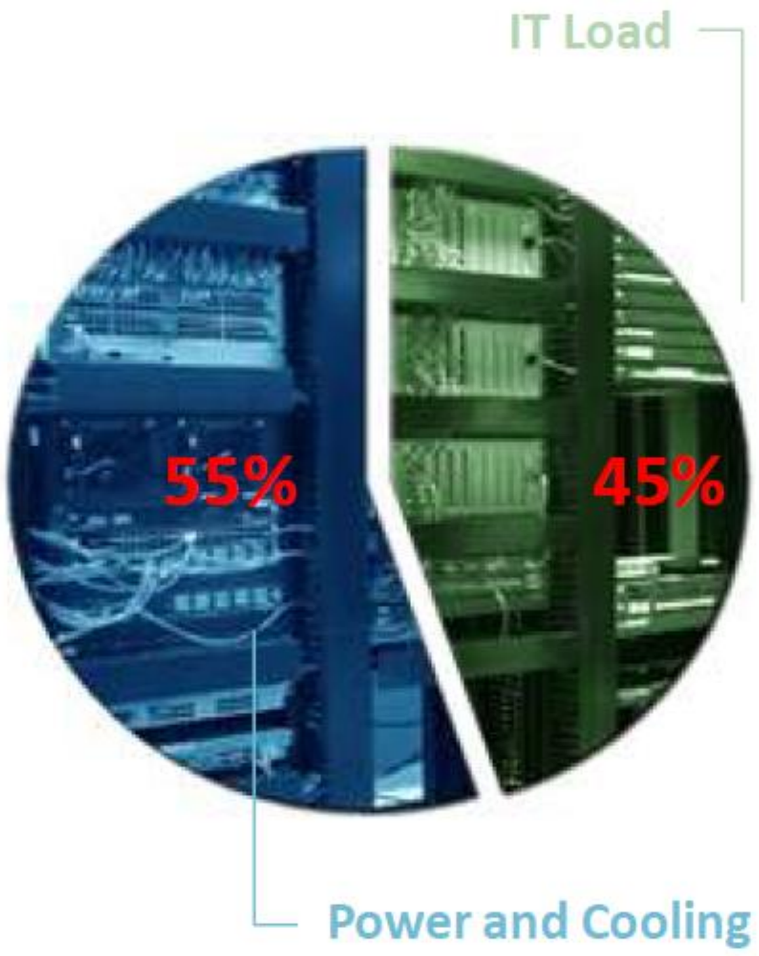
Case study: Cloud


Cloud scheduling taxonomy
Application layer
Platform layer
Infrastructure layer
Scheduling w.r.t. user QoS
Scheduling w.r.t. provider efficiency
Scheduling w.r.t. negotiation
Scheduling w.r.t. load imbalance
Energy aware scheduling
Scheduling w.r.t. cost
Scheduling w.r.t. service assignment
Scheduling w.r.t. partner federation
Scheduling w.r.t. data communication

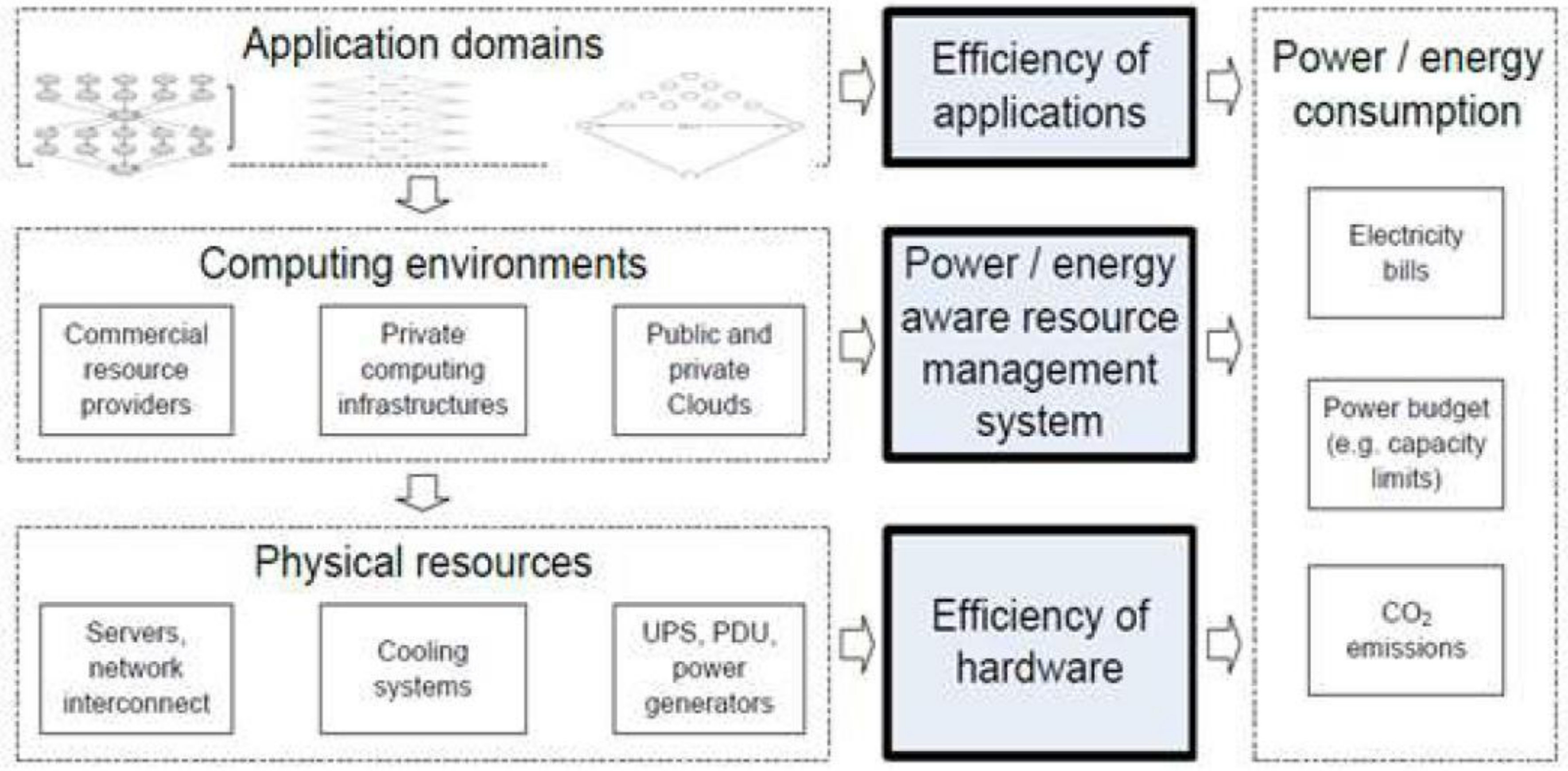
Why energy is an issue?

- Traditionally performance have been the main interest in system design and development
- With energy price souring and environmental concerns, energy consumption management has become an important issue in various domains.
  - – Exascale systems (HPC)
  - – Cloud data center
  - – IoT devices (e.g., portable medical devices)
  - – Embedded systems - Mobile and portable devices (e.g., digital camcorders, mobile phones), laptops
  - – Sensor network applications

How is the energy typically used in a data center



Energy consumption at different levels


Application domains
Computing environments
Commercial resource providers
Private computing infrastructures
Public and private Clouds
Physical resources
Servers, network interconnect
Cooling systems
UPS, PDU, power generators
Efficiency of applications
Power / energy aware resource management system
Efficiency of hardware
Power / energy consumption
Electricity bills
Power budget (e.g. capacity limits)
CO_2 emissions

Power vs. energy

- Power is the rate at which the system performs the work,
- Energy is the total amount of work performed over a period of time.

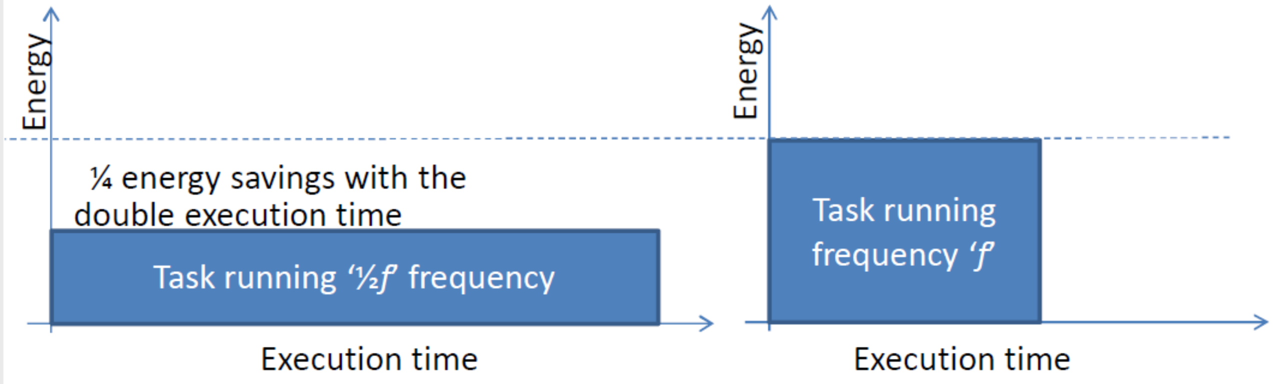
  If P is power, T is a period of time, W is the total work performed in that period of time:

  $$P=W/T,\ E=PT$$

- A reduction of the power consumption does not always reduce the consumed energy

Lowering power

- Running a task at a slower speed saves power


Energy
¼ energy savings with the double execution time
Task running '½f' frequency
Execution time
Energy
Task running frequency 'f'
Execution time

- The problem is that it will lake longer to finish the task thus affecting the performance
- What if we could reduce the energy used with minimal performance impact?

Dynamic power management

- Dynamic Power Management (DPM) is a design methodology for energy and power management of dynamically reconfiguring systems.
- The goal for a DPM system is to provide the requested services and performance with a minimum power consumption.
- An example of DPM is the ‘Dynamic Voltage and Frequency Scaling (DVFS).’

Cost of Cloud resources

- Cloud users pay providers for cloud resource usage.
- For example, let $\mathcal{R} = \{r_1, \cdots, r_n\}$ be the resources used by Cloud users.

$$U_i = \sum_{i \in \mathcal{R}} c_{r_i} \cdot t_i$$

  – c_i denotes the cost of resource r_i per unit time and t_i denotes the time for which resource r_i is utilized.

- For example, if the cost of using resource r1 is 2000 per time unit and r2 is 3000 per time unit, where r1 is used for 22 time unit and r2 is used 19 time unit
  - $U_i = (2000 \cdot 22) + (3000 \cdot 19) = 101{,}000$

Quality of service (QoS)

- Cloud users expect Quality-of-service guarantees from the cloud service providers.
- QoS parameters indicate the ability of a service to meet certain requirements for different aspects of the service
- For example

  – Deadline constraint: This represents the time till which the task or the batch of tasks should be finished.

  – Budget constraint: This represents the restriction on the total cost of executing all tasks.

  – Cost: resources are provisioned in a cost-efficient way

User objectives

- User objective can be stated as maximizing QoS such as minimize the usage cost while satisfying the performance requirement of the applications
- For example, the objective can be to determine a configuration where all tasks of a user are executed at a minimal cost.

$$\text{Minimize } \left(\sum_{i \in \mathcal{R}} c_{r_i} \cdot t_i\right)$$

- QoS requirements are commonly formalized in the form of SLAs

Cloud service provider objectives

- Cloud providers need to efficiently manage resources to achieve the performance of their applications and improve the utilization of reserved resources, thereby minimizing the usage cost.
- Some objectives
  - – Revenue maximization
  - – Resource utilization maximization

  These must be done in measured manner is it can lead to system overload.

How to achieve both side objectives?

- Cloud providers must achieve certain level of QoS
- For instance
  – For budget constraint jobs the cost of performing these jobs cannot exceed a certain budget constraint
- Therefore, one of the challenges facing service provider is how to complete jobs with unpredictable submission time under budget and deadline constrains
  - Most of existing work do not consider unpredictable submission time of jobs, as well as budget and deadline constrains simultaneously

Service Level Agreements (SLA)

- Service Level Agreements (SLAs) form an important component of the contractual relationship between a cloud customer and a cloud service provider
- Different cloud services and deployment models will require different approaches to SLAs
- The global nature of Cloud computing renders SLAs to cross several jurisdictions
- Frameworks for handling data privacy is an important addition to cloud computing
- Total economical penalties for SLA violations the sum of the total proportional penalties costs for unsatisfied demand of resources

SLA-aware scheduling

- The objective is to satisfy the SLA requirements of the user while minimizing the total cost
- If SLA requirements consist of Budget and Deadline, the algorithm objective

$$\text{Minimize}\left(TC = \sum_{i \in \mathcal{R}} c_{r_i} \cdot t_i\right)$$

- Subject to
  - $Total\ Cost\ \leq budget$
  - $execution\ time\ \leq deadline$

Optimization algorithms

- Optimization algorithms can be roughly divided into two categories: exact algorithms and heuristics.
- Exact algorithms are designed in such a way that it is guaranteed that they will find the optimal solution in a finite amount of time.

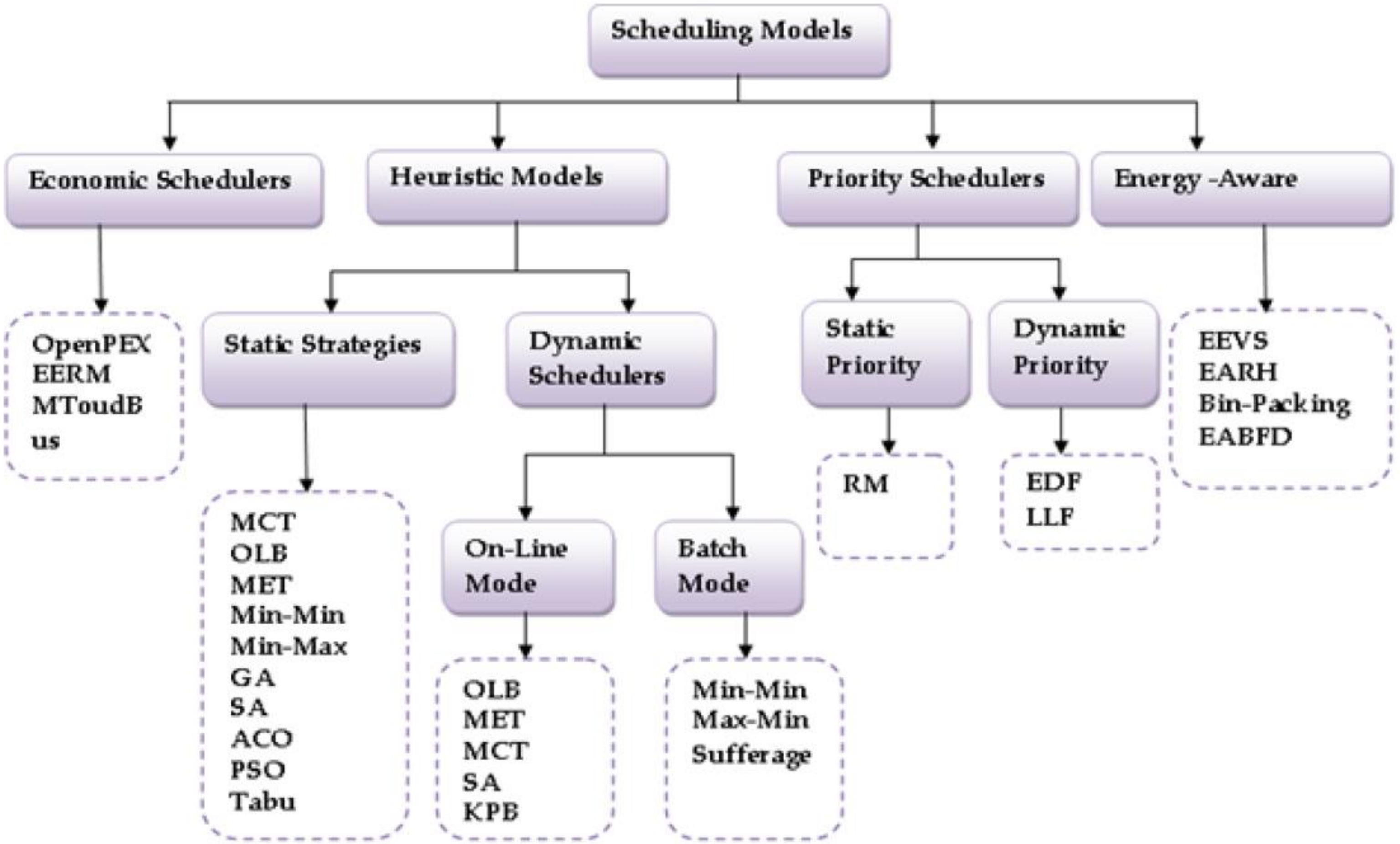
  –For scheduling optimization problems (e.g. NP-hard or global optimization) this "finite amount of time" may increase exponentially in respect to the dimensions of the problem.
- Heuristics do not have this guarantee, and therefore generally return solutions that are worse than optimal.

  –Heuristic algorithms usually find "good" solutions in a "reasonable" amount of time

  - adapted to the problem at hand and they try to take full advantage of the particularities of this problem
  - because they are often too greedy, they usually get trapped in a local optimum and thus fail, in general, to obtain the global optimum solution.

  Heuristic algorithms are generally make simplifying assumptions to relax constrains.

A general view of the solutions



Examples of Rule-based Heuristics for Clouds

- Min-Min
- Sufferage
- Max-Min
- Minimum Completion Time (MCT),
- Minimum Execution Time (MET)
- First Come First Serve (FCFS)
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)
- Bin Packing
- Genetic Algorithm

Min-Min Heuristic

- Min-Min heuristic initiates with the set MT [Meta-Task] comprising of all unassigned tasks and executes in two phases.
- 1st phase establishes minimum expected completion time (for each task in MT) on each machine.
- 2nd phase chooses the task boasting minimum expected completion time among the set of tasks in MT.
- Further, the chosen task is assigned to the matching resource (having minimum expected completion time).
- Finally, the assigned task is removed from MT and the process is replicated multiple times until each task in the MT is mapped

Key Notations

Symbol	Definition
MT	Meta-Task comprising of submitted tasks
VMT	Set of virtual machines in data center
Ti	Current task in Meta-Task
VMj	Current vm instance in VMT
Tv	Task with Maximum Completion Time
SVi	Sufferage value for task Ti
Ts	Task with maximum sufferage value
MCTi	Minimum Completion Time of Task Ti
Sec_MCTi	Second Minimum Completion Time of Task Ti
Tu	Task with Minimum Completion Time
VMv	Virtual Machine that takes minimum completion time

Min-Min Heuristic

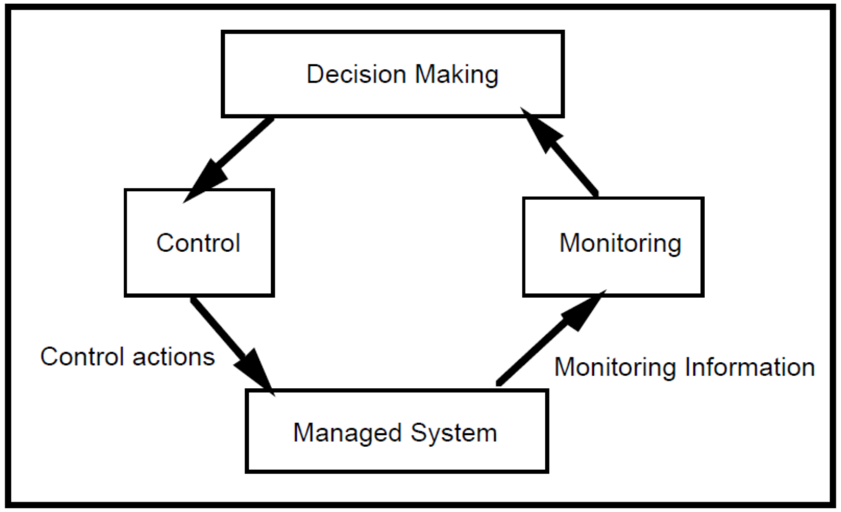
```
Input: Meta-Task MT, Task Length Mi, Resource Speed MIPSj, Resources VMT, Execution Time Matrix ETi
Output: Mapped Schedule S:S(T1), S(T2),…..S(Tn)
Begin:
MT←{T1,T2,….Tn), VMT←{VM1,VM2,….VMm)
While MT ≠ $\phi$ Do
        For Ti ∈ MT Do
                For VMj ∈ VMT Do
                        Compute ETi,j = Mi / MIPSj
        For Ti ∈ MT Do
                For VMj ∈ VMT Do
                        Compute CTi,j= ETi,j + Rj
        Find MCTi // Tu on VMv,
        Sort(CTi,1,CTi,2,…..CTi,n)
        Find S(Tu) for (Tu, VMv)
        Compute MT=MT- Tu //Delete task Tu from Meta-Task
        Ru=CTu,v // Update ready time of machine v
        For Ti ∈ MT Do CTi,v=Ei,v + Rv
End
```

Resource Monitoring

- Is a key tool that provision controlling and managing software and hardware infrastructures
- It bestows and maintains key performance indicators that facilitate data collection to aid in decisions associated with resource allocation process.
- It monitors state of resources at the time of failure at physical or service layers.

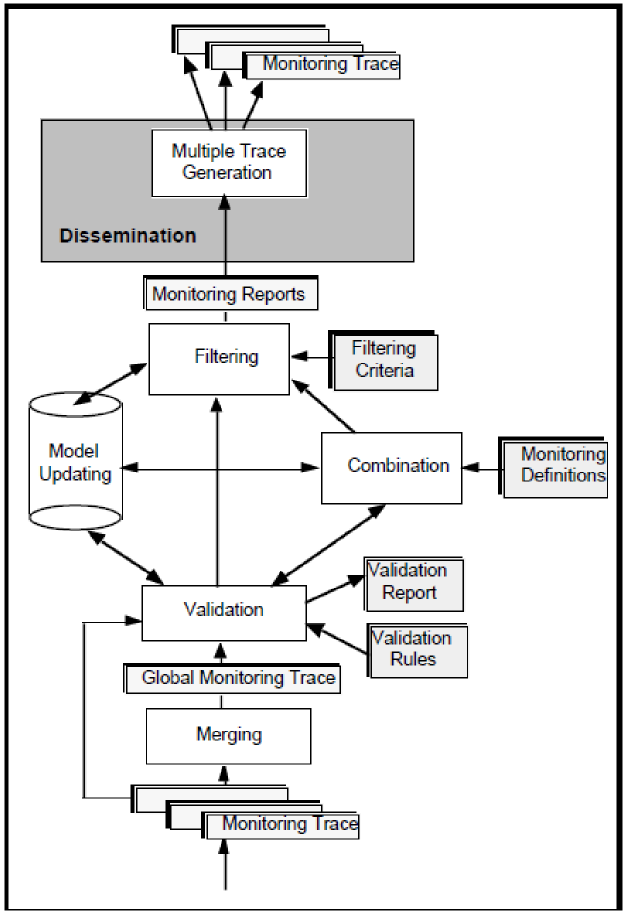
Monitoring reports

Are used by human or automated managers to make management decisions, resulting in control actions that may modify the behaviour of the managed systems



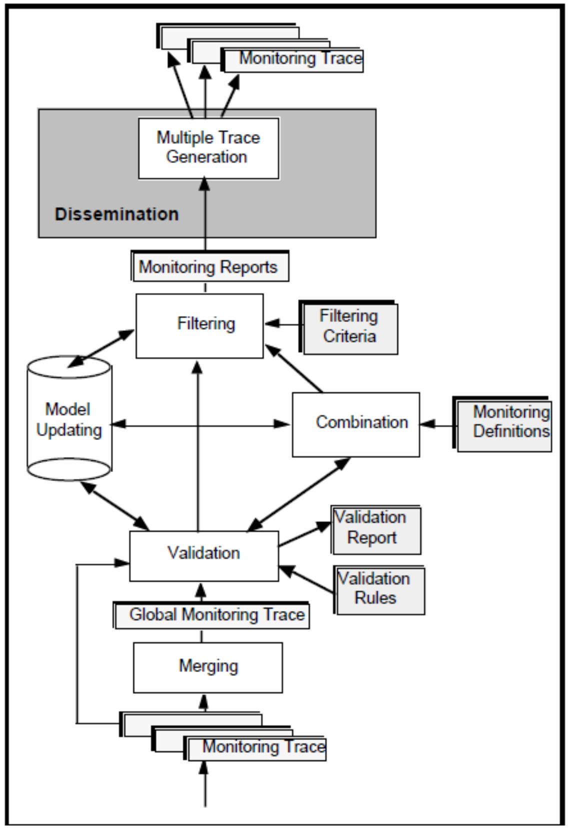
Elements of a monitoring reference model (1/2)

1. Generation of monitoring information
   - Status reporting
   - Event detection and reporting
   - Trace generation
2. Processing of monitoring information
   - Merging and multiple trace generation
   - Validation
   - Model updating
   - Combination
   - Filtering
   - Analysis
3. Dissemination of Monitoring Information
   - Registration of subscribers to dissemination service
   - Specification of information selection criteria



Elements of a monitoring reference model (2/2)

4. Presentation of Monitoring Information
   - Textual displays
   - Time process diagrams
   - Animation of events and status
   - User control of levels of abstraction
   - User control of information placement and time frame for updates
   - Multiple simultaneous views
   - Visibility of interaction message contents
5. Implementation Issues
   - Special purpose hardware
   - Software probes
   - Time synchronisation for event ordering



Time-driven monitoring vs. Event-driven monitoring

- Time-driven monitoring is based on acquiring periodic status information to provide an instantaneous view of the behaviour of an object or a group of objects.
  - There is a direct relationship between the sampling rate and the amount of information generated.
- Event-driven monitoring is based on obtaining information about occurrence of events of interest, which provide a dynamic view of system activity.
  - The amount of generated and communicated monitoring data is reduced as only the information pertaining to activity of interest is collected and transmitted.
  - Most common approach adopted in monitoring systems.
    - These systems provide automatic recognition of events which means that the user need not explicitly collect and analyse the lower level details of system behaviour

Cluster vs. Grid vs. Cloud Monitoring

- Monitoring of Cloud is more complex due to the trust model and the view on resources/services presented to the user
  - Grid: simple accounting criteria and limited respurce abstraction leading to simple relation between monitoring parameters and physical resource status
  - Cloud: high abstraction of resources leading to opaque relationship between the layer- or service- specific observables and underlying resources
- Most of the monitoring approaches/platforms proposed for the Grid case have been customized for Cloud sustems
  - Grid: Ganglia, Nagios, MonaLisa, R-GMA, GridICE
- Clusters comparable to a base technology for Cloud IaaS Providers
- Most properties of Cloud monitoring systems do not apply to Cluster/Grids (e.g. elasticity, adaptability, autonomicity) or are not vital (e.g. extensibility, intrusiveness)

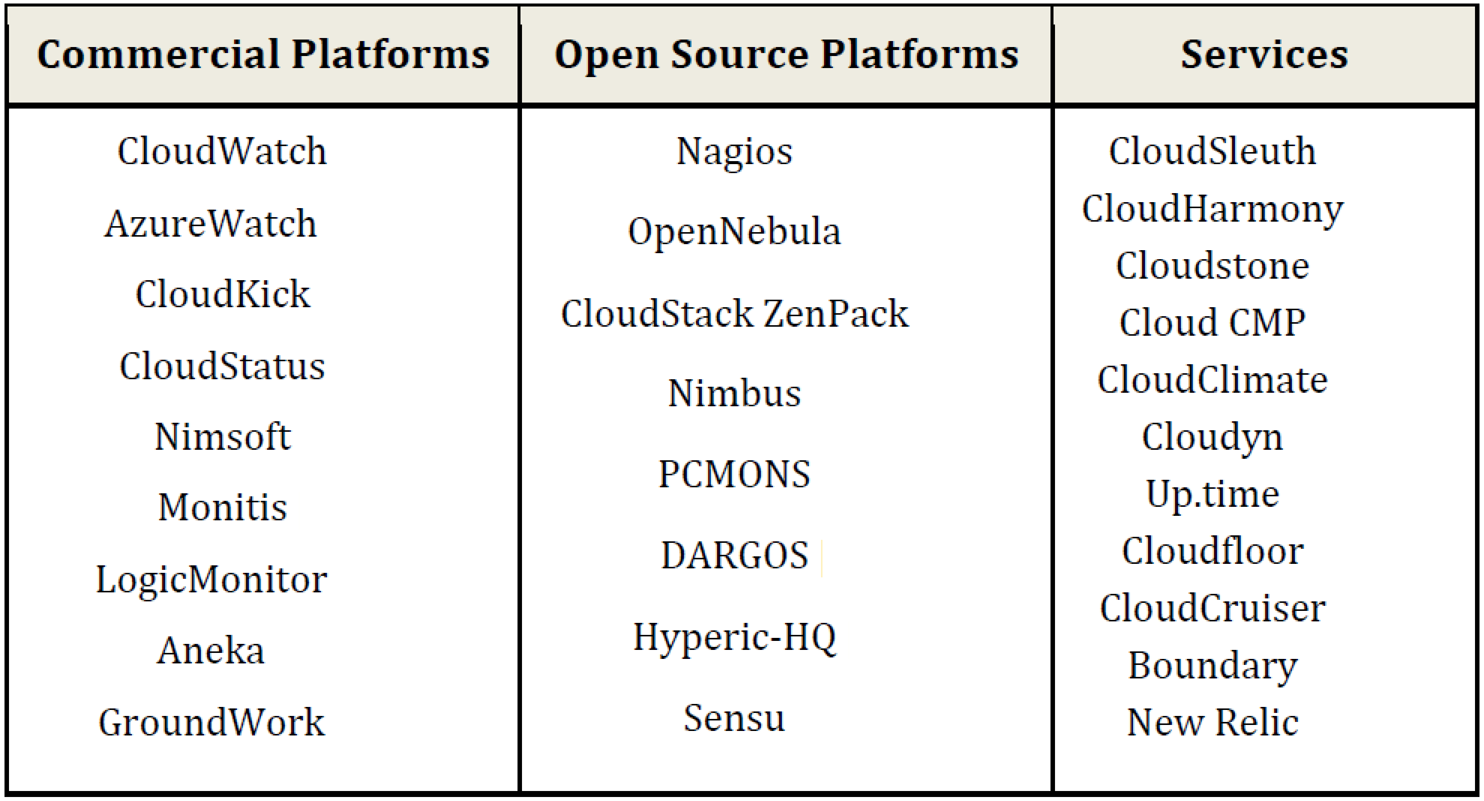
Metrics in computation-based tests

- server throughput (no. requests/second),
- CPU speed,
- CPU time per execution (CPU time of a single execution),
- CPU utilization (CPU occupation of each virtual machine),
- memory page exchanges per second (no. memory pages/second exchanged through the I/O);
- memory page exchanges per execution (no. memory pages used during an execution);
- disk/memory throughput;
- throughput/delay of message passing between processes;
- duration of specific predefined tasks;
- response time;
- VM startup time;
- VM acquisition/release time;
- execution/access time,
- up-time

Metrics in network-based tests

- round--trip time (RTT),
- jitter,
- throughput,
- packet/data loss,
- available bandwidth,
- capacity,
- traffic volume

Cloud monitoring platforms and services



Commercial Platforms	Open Source Platforms	Services
CloudWatch	Nagios	CloudSleuth
AzureWatch	OpenNebula	CloudHarmony
CloudKick	CloudStack ZenPack	Cloudstone
CloudStatus	Nimbus	Cloud CMP
Nimsoft	PCMONS	CloudClimate
Monitis	DARGOS	Cloudyn
LogicMonitor	Hyperic-HQ	Up.time
Aneka	Sensu	Cloudfloor
GroundWork		CloudCruiser
		Boundary
		New Relic