**General Notes**

**Introduction**

* Large virtualized data centers (DC) serve growing demand for
  + Computation
  + Storage
  + Networking
* Energy consumption is fastest-growing part of energy consumption of ICT ecosystem
* Initial costs for purchasing equipment for DC outweighed by cost of ongoing electricity consumption
* DC extensive use of virtualization technology 🡪 ensure isolation of applications while allowing healthy utilization of physical resources
* Options to save energy in DCs:
  + Consolidate the VMs to minimal number of physical hosts and switching unused hosts off

🡪 Downsize: aggressive VM consolidation can lead to overloaded hosts with negative effects on delivered quality of service (QoS) + violating SLAs with customers

🡪 Result: VM allocation must find optimal balance between QoS and energy consumption

* + Switch to less power-hungry mode of operation (sleep mode)
* Focus of paper: VM allocation problem
  + Example: Determining the placement of VMs on physical hosts or using external providers
  + considering QoS guarantees, costs associated with using the hosts (emphasizing energy consumption and penalties as result of VM migrations
* Problem: all available algorithms addressing the VM allocation problem focus on different versions (aspects) of it by differing
  + In the way of the communication between hosts is modeled
  + How multicore CPUs are handled
* There is no general definition of VM allocation problem 🡪 every available research in this area gives a different definition of it based on the focus of the corresponding research

🡪 Impact: differences have substantial impact on algorithm runtime and/or on applicability of the algorithm

**Problem context**

[Theory]

* VM allocation problem one of the core challenges of using the cloud computing paradigm efficiently

🡪 depending on cloud computing setup allocation problem varies

* Two dimensions of cloud computing scenarios
  + 1st dimension: nature of offered service differentiating between IaaS, PaaS and SaaS
  + 2nd dimension: focus whether service is provisioned in-house (private) or public or combination of two (hybrid)
* Other classification of cloud computing focuses on service deployment scenarios
  + Assumption: service provider (SP) deploys a service on the infrastructure provided by one or more infrastructure providers (IP)
* Statement: in each scenario there is need to optimize allocation of VMs to physical resources

🡪 optimization may be performed by different actors and might have different characteristics depending on exact setup

* Based on Li et al. classification VM allocation problem occurs as following:
  + Public cloud: IP must optimize utilization of resources to find best balance between conflicting requirements
  + Private: same optimization problem occurs for provider that act as SP and IP
  + Bursted:
    - IP must sole same kind of optimization as above
    - SP must solve similar problem for own resources extended by possibility to off-load some VMs to external IP
  + Federated: IP must solve optimization problem similar to SP
  + Multicloud:
    - IP must solve same optimization problem as in public cloud
    - SP must solve optimization problem where optimal allocation of parts of service to IP is defined
  + Cloud broker: same as multicloud scenario where broker owns role of SP
* Assumption: cloud provider (CP) must allocate VMs to a set of available resources

🡪 resources either belong to CP or rented from external cloud providers (eCP)

**Problem characteristics**

* Main characteristics of VM allocation problem:
  + CP accommodates VM on physical machine (PM) or by renting capacity
  + Number of VM changes over time as result of upcoming requests to create additional VM or remove VM
  + Resource requ. Of VM can vary over time (CPU, RAM, storage, network communication)
  + PM has limited capacity in terms of resources
  + Use of resources incurs monetary costs and consumes electric power
  + VM can be migrated from one PM to another by means of live migration 🡪 creates additional load for involved PM and network
  + PM not used by any VM can be switches to low-energy state
  + Penalty if QoS requ. Of customer is not met

Live Migration

* Definition: migration of a VM from one PM to another
* Advantages:
  + allows to react to changing resource requirements of VM
  + in times of low demand: several VMs can be consolidated to one PM so that other PMs can be switched off including saving energy
* Downsize:
  + VM migrations take time
  + Create overhead
  + Can have adverse impact on SLA fulfillment
  + VM migration may increase load of source and target PM
  + Put additional burden on network and make migrated VM less responsive during migration
* Solution to avoid downsize: keep number of live migrations at reasonable level

**Actions of CP**

* Cases where CP has to update VM placement:
  + Reaction to customer request
  + Reaction to critical situations and changes in system load
  + In course of regular evaluation of current placement to improve overall optimization objectives
* Reoptimizing of VM placement consists of one or more of the following actions:
  + Migration of VM from one host to another
  + Switching state of PM
  + Starting/Ending rental of VM from eCP
  + VM resizing

**Scientific Research**

* Single-DC problem: CP has single DC with number of PM and no eCP
  + Typical objectives:
    - optimizing the utilization of resources and minimizing overall energy consumption
    - subject to performance constraints (by SLAs)
  + all PMs in same DC 🡪 network bandwidth often assumed to be uniform and sufficiently high so that it can be ignored
* Multi-IaaS problem: CP does not own any PMs and uses leased VM from multiple IaaS providers
  + Since no PMs: concerns related to PMs, states and state transitions, sharing of resources among multiple VMs, load-dependent power consumption are void
  + Power consumption plays no role
  + Main goals:
    - Minimizing monetary costs associated with VM rental and maximizing performance
    - Bottleneck of data transfer between different IaaS providers
* One-dimensional VM placement problem: often-investigated special case 🡪 only computational demands and capacities are considered

🡪 CPU seen as single core to make problem one-dimensional

🡪 can be special case of single-Dc, multi-IaaS or other problem formulations

* On/Off problem: each PM only has two states (On/Off)
  + Core elements:
    - Power off: power consumption is 0
    - Power on: power consumption is positive constant for each PM
    - Dynamic power consumption not considered
    - Transition between two states assumed to be instantaneous

🡪 Focus: minimize number of PMs that are on

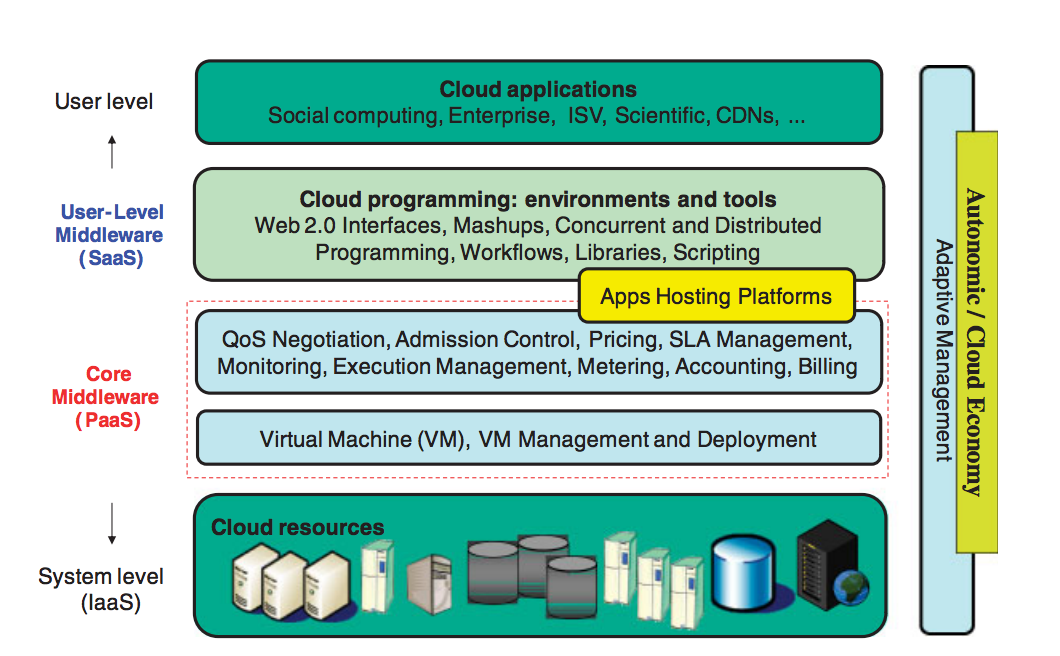
* Online vs Offline Optimization: ways how CP reacts to customer requests
  + Online optimization: finding best reaction to customer requests in given situation
  + Offline optimization: finding best new configuration
  + For each optimization CP may use two different algorithms
* Placement Tasks
  + Initial placement: determines placement for new set of VM
  + Placement reoptimization: optimizes existing placement and must use migrations
* Load Prediction Problem: when CP makes some change in mapping of VM or states of PM
  + Change at time t0: decision can only be based on observations of VM behavior for period t < t0

🡪 decision will have effect for t > t0

🡪 CP can only make ideal decision if the future resource utilization of VM is known

🡪 since future unknown: important subproblem to predict resource utilization values of VM or probability distributions at least for near future

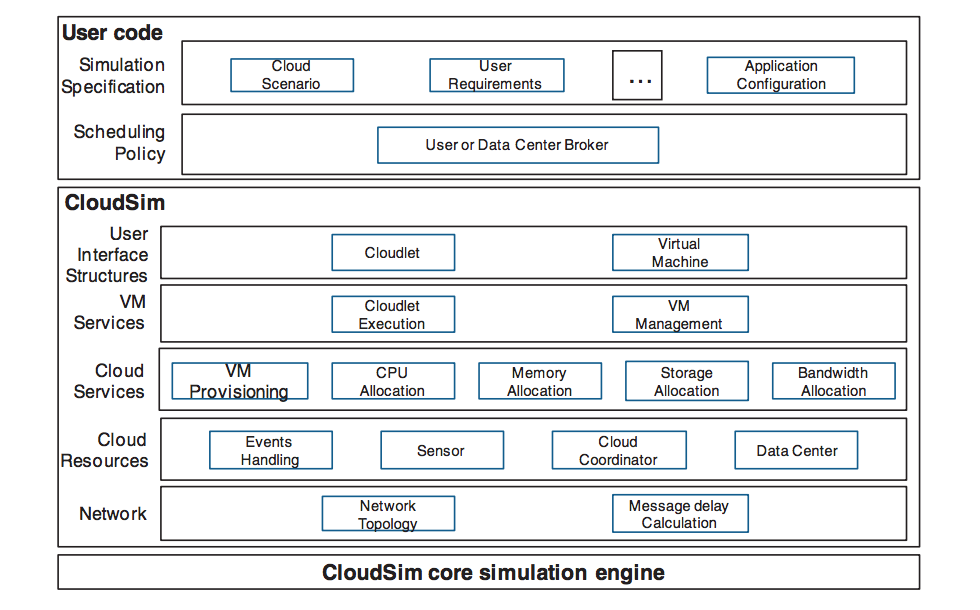
**Cloud computing architecture**



* Cloud applications
  + applications directly available to end-users 🡪 applications can be supplied by Cloud provider (SaaS) or user deploys own application
* User-Level middleware
  + Includes software frameworks e.g. Web 2.0 interfaces to help developers to create rich, cost-effective user-interfaces for browser-based applications
* Core middleware
  + Implements platform-level services that provide run-time environment for hosting and managing user-level application services
* System level
  + Computing power in cloud environments provided by collection of data centers that are installed with hundreds to thousands of hosts

**CloudSim General**

* Event driven simulator built on top on core of grid simulator GridSim
* Supports several core functionalities, e.g. queuing and processing of events, creation of CloudSim entities, communication among components and management of simulation clock
* Layered cloudsim architecture



* Defines parameters of cloud environments such as hosts, VMs, applications and datacenters by instances of different classes
* Datacenter: resource provider which simulates infrastructure as a service

🡪 multiple hosts are created inside datacenters

🡪 in CloudSim: at least one datacenter must be available to start execution

* DatacenterBroker: responsible for application scheduling and coordinating the resources

🡪 datacenter broker functions like coordinating entity of resources and user applications

🡪 single broker or hierarchy of brokers can be initiated depending on simulation scenario

* CPU unit is defined by Processing Element with million instructions per second

🡪 processing elements are shared resources

* Cloudlets: represent applications that share resources (CPU) among them

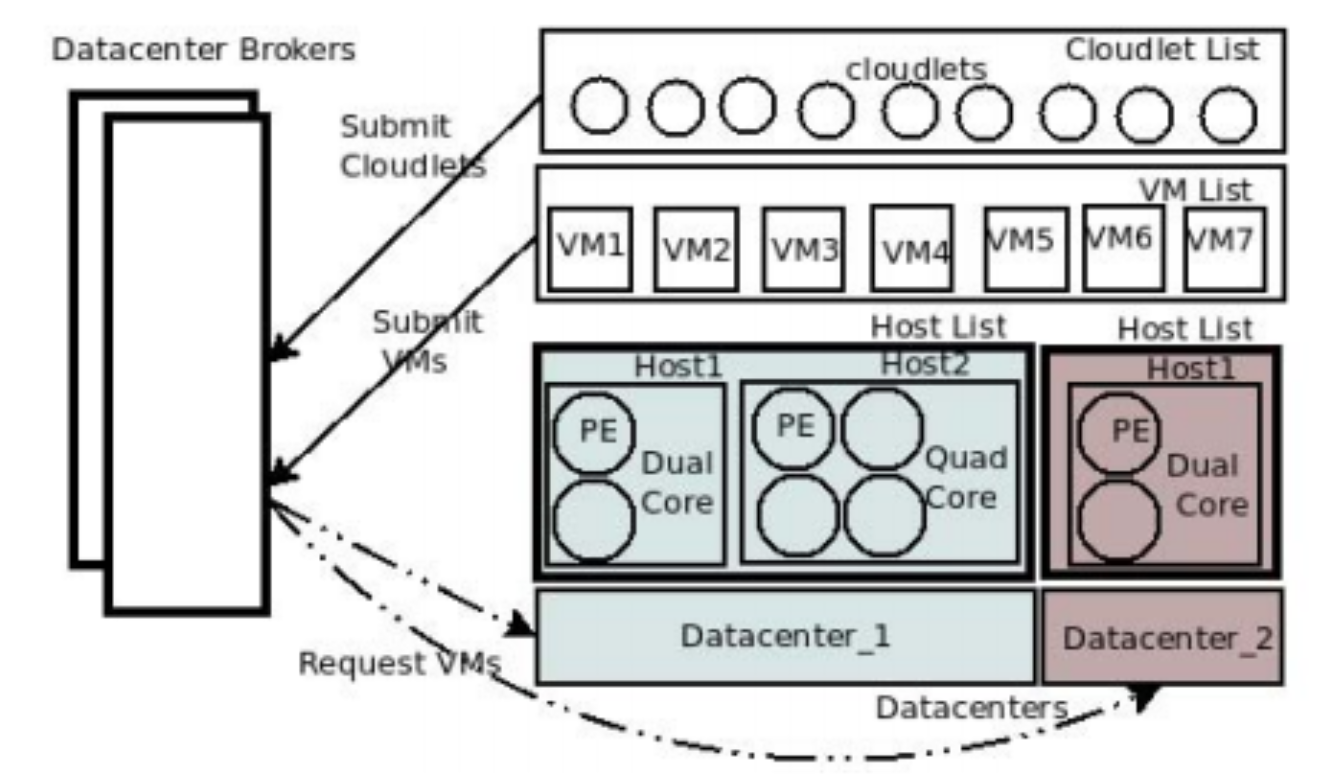
🡪 status: FREE, BUSY/Allocated or FAILED (indicates availability of cloudlet)

* Dependency: each VM assigned to a host where each cloudlet is assigned to VM

🡪 processing elements shared among VMs in host and executing cloudlets in VMs

* When simulation is started the list of cloudlets and VMs is sent to broker

🡪 broker handles allocation of VMs to hosts and cloudlets to VMs and leads the simulation behavior such as deciding which available cloudlets should be executed next



**Branch and Bound Algorithm**

* Developed by Narendra and Fukunaga in 1977
* Helps to identify parts of search space that can be left unexplored without missing optimal solution
* Guaranteed to find optimal feature subset without evaluating all possible subsets
* Exponential search method
  + Algorithm used for searching sorted, unbounded/infinite arrays
  + Determines a range that the target value resides in and perform a binary search within that range
  + Assuming the array is sorted in ascending order it searches for the first exponent k where the value 2k is greater than search key
* B&B algorithm based on finding optimal solutions for given problem by means of two procedures called branching and bounding
* Branching procedure: splitting original solution set into subsets that are easier to solve
* Bounding procedure: determine which subsets will be expanded (branching operation) and which ones will be bound (bounding operation)
* Bounding process aims to reduce number of generated subsets in B&B tree by calculating lower and upper bounds based on current solution
* Operations are applied iteratively to active subsets and unpromising subsets can be eliminated thus search space is reduced
* Using B&B algorithm enabled to determine the bounds as tight as possible in addition to finding an optimal solution of the problem so that computation time is kept at minimum