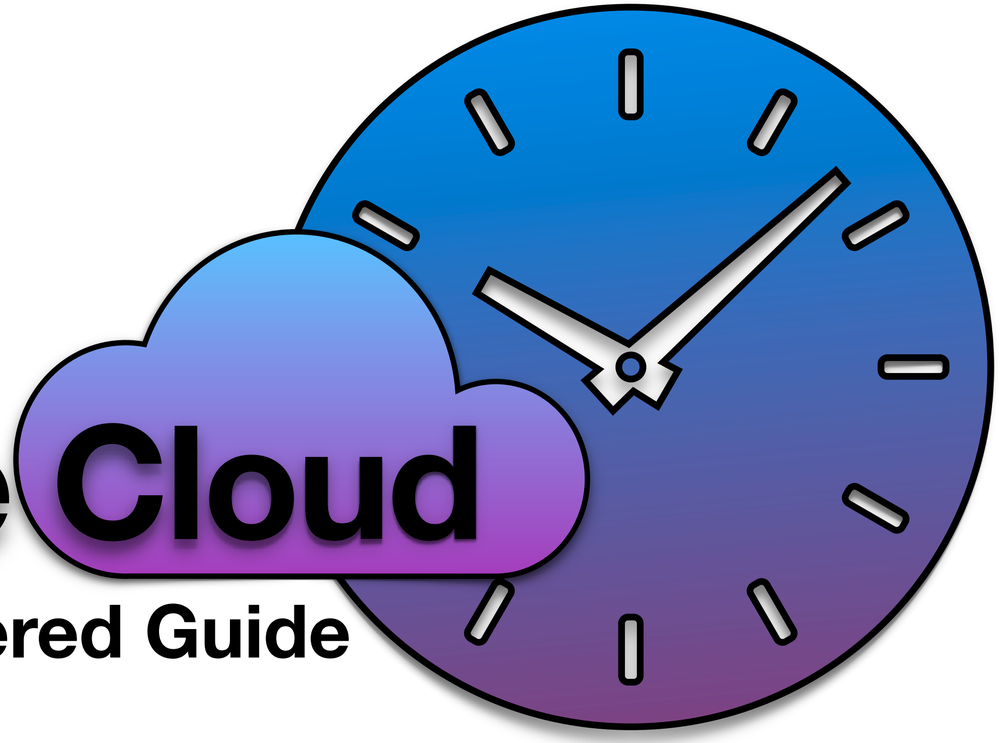
Scheduling in the Cloud



A Theoretical and Simulation-Centered Guide

Parsa Toopchinezhad - Laya Hamekhani - Mahshid Azizi | Cloud Computing | Fall 2024

Table of Contents

I

Introduction

- Importance of scheduling
- Scheduling in cloud computing
- Objectives of scheduling
- Space V.S. time-shared scheduling

II

Theory

- Exact methods
- Heuristic algorithms
- Meta-heuristic algorithms
- Learning-based algorithms

III

Simulation

- Importance of simulations
- Introduction to CloudSim
- CloudSim installation
- Algorithm implementation and comparison

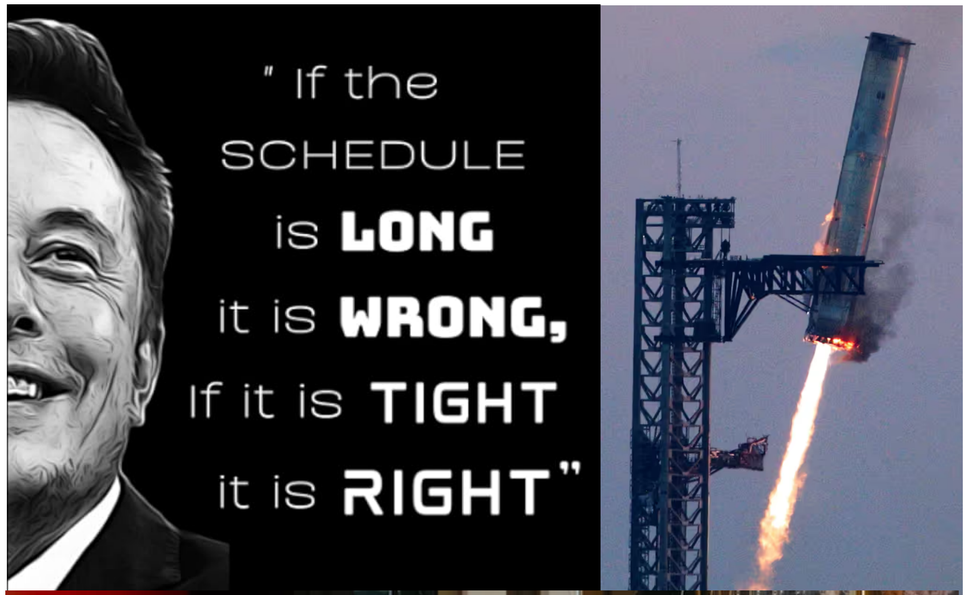
Learning Objectives

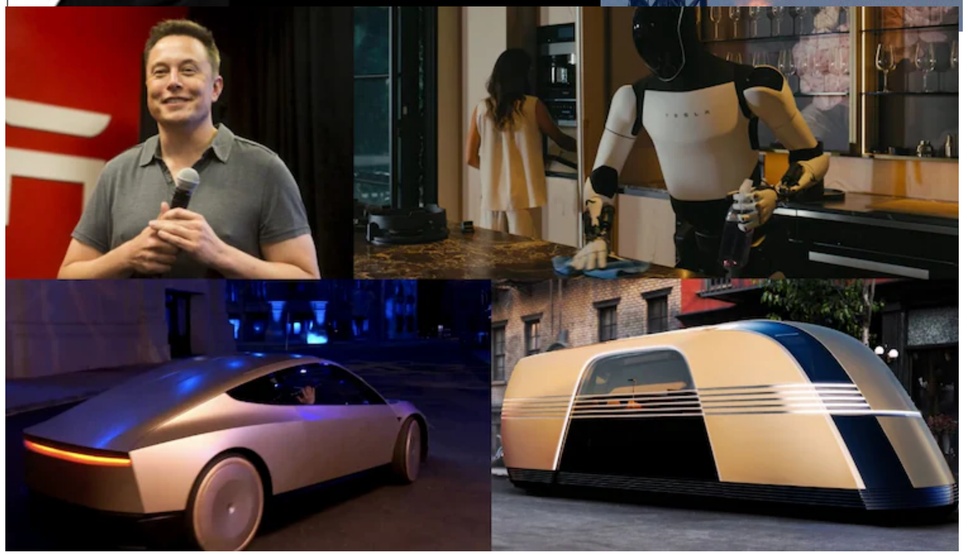
By the end, you should know:

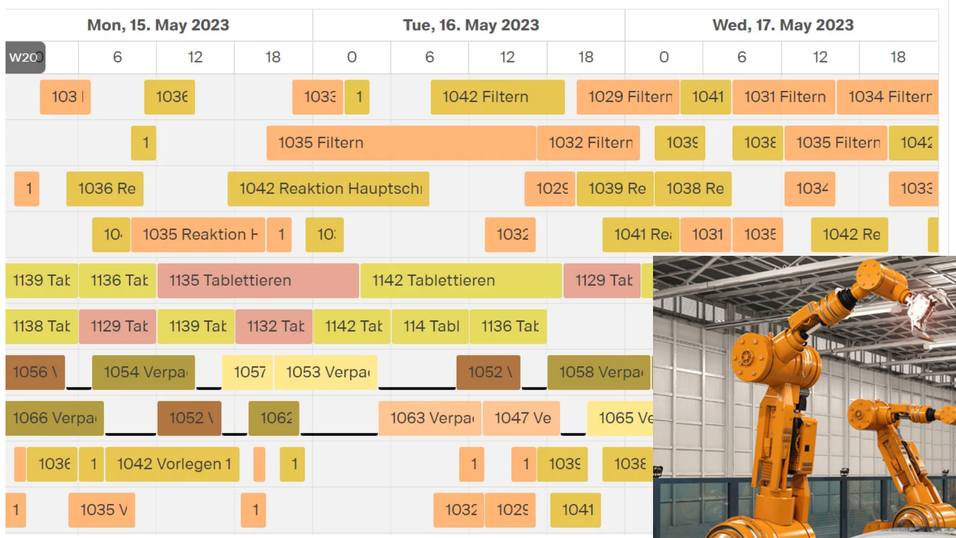
- Why scheduling is a crucial and non-trivial problem
- The major different approaches to scheduling
- The advantages and limitations of scheduling algorithms
- How simulations help researchers verify new ideas
- How to use CloudSim to test a novel algorithm
- The current direction of scheduling research

Part I: Introduction

Scheduling: An Ever-Present Optimization Problem

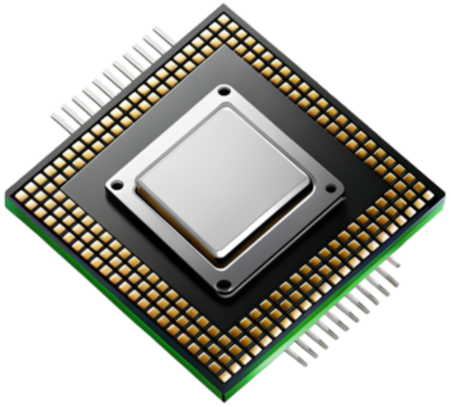






Linux Scheduler History

- Linux v1.2 – Round Robin
- Linux v2.2 – Scheduling Classes & Policies
- Linux v2.4 – Division in epochs, goodness of function
- Linux v2.6 – Runqueue O(1)
- Linux v2.6.21 – Completely Fair Scheduler (CFS)

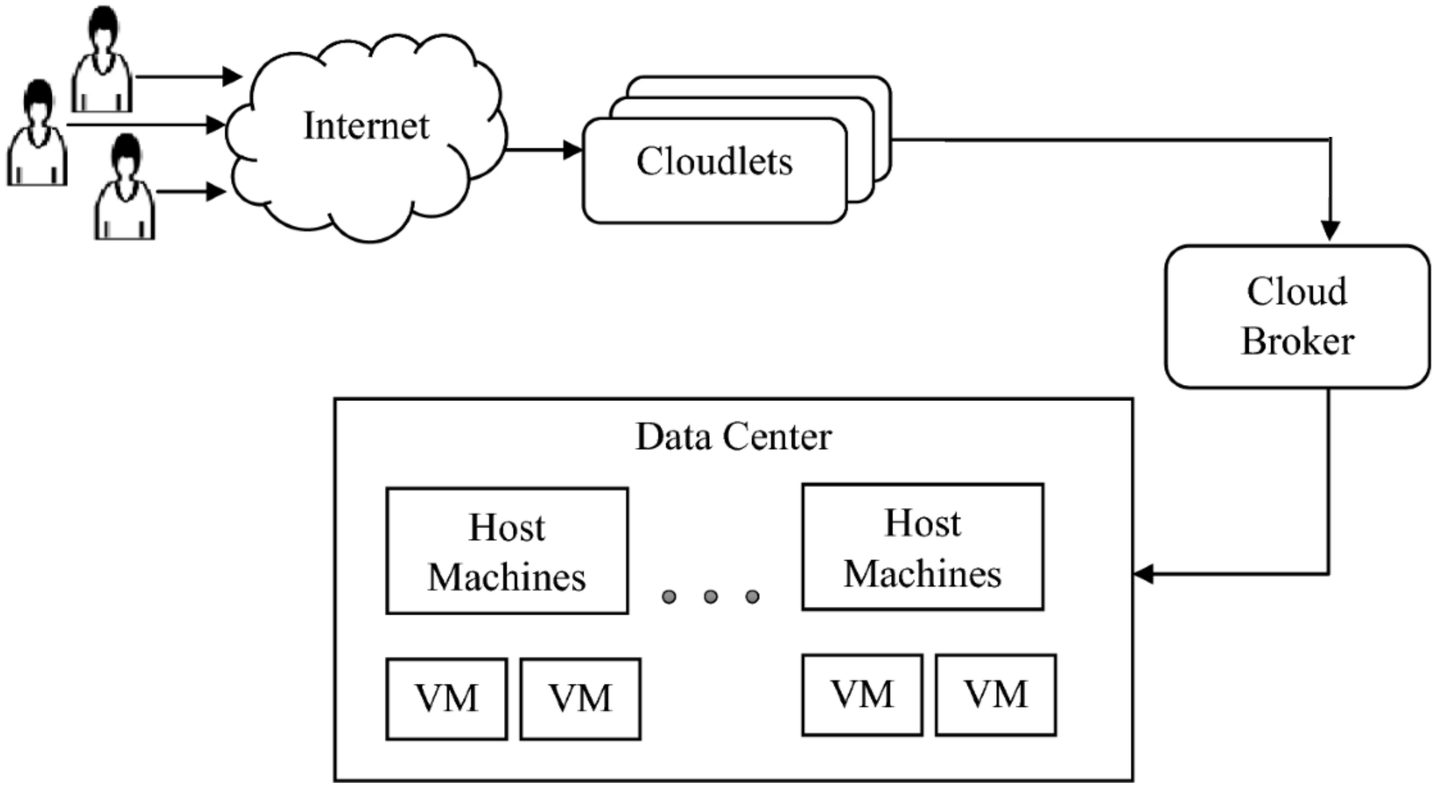


Task Scheduling

An essential part of cloud computing

Important Terms

- **Cloudlet**: tasks generated by users
- **Broker**: middleware that assigns tasks to VMs via a scheduling algorithm
- **Host/Physical Machine**: the hardware hosting the VMs
- **Datacenter**: a collection of (at least one) host machines

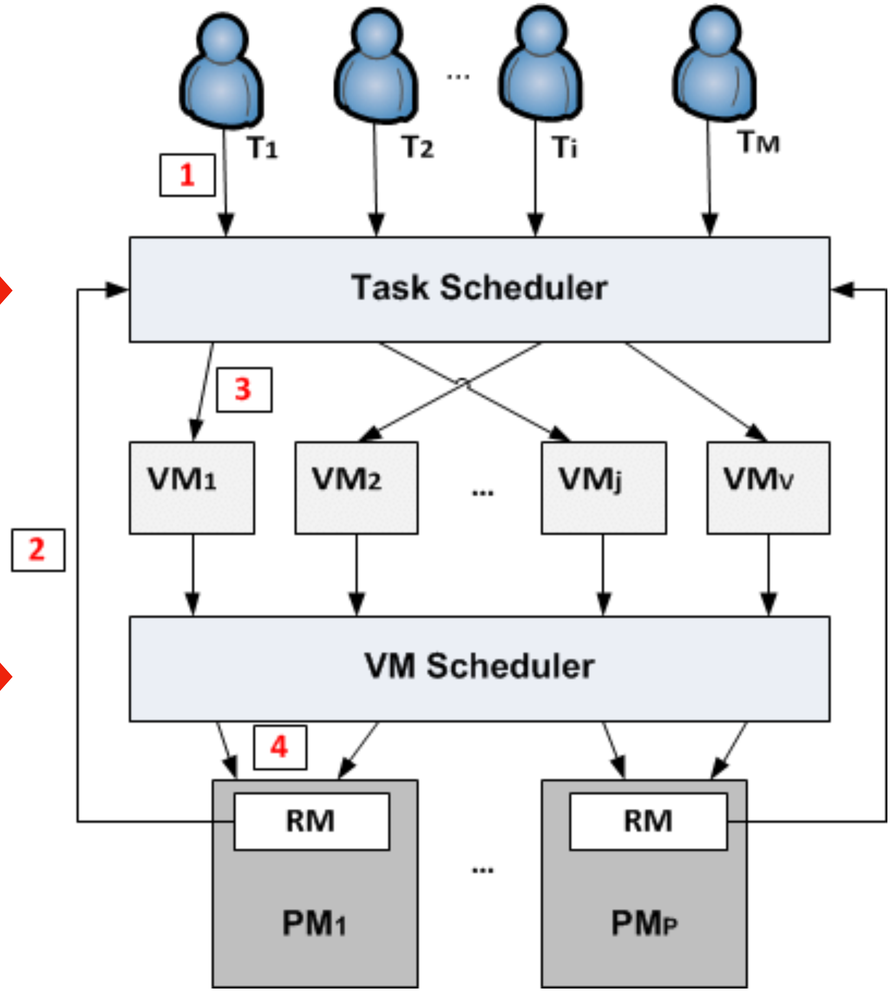


Task Scheduling (Cont.)

The two levels of scheduling

We will be focusing on this problem today

However, this is also an interesting problem



Why Scheduling Matters

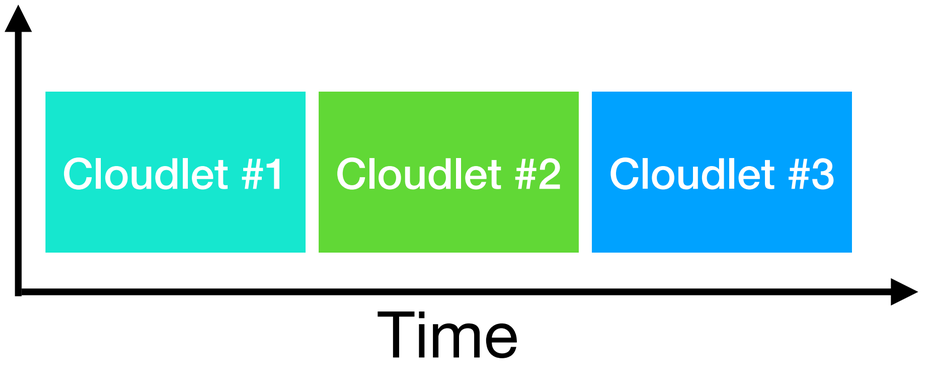
The many possible objectives

- **Resource Utilization** ———> Ensure nothing is being wasted
- **Load Balancing** ———————> Ensure performance remains consistent
- **Minimizing Latency** ———> Increase QoS & ensure SLA
- **Task Priority Management** —> Ensure critical task deadlines are met
- **Energy Efficiency** ———> Save energy & money

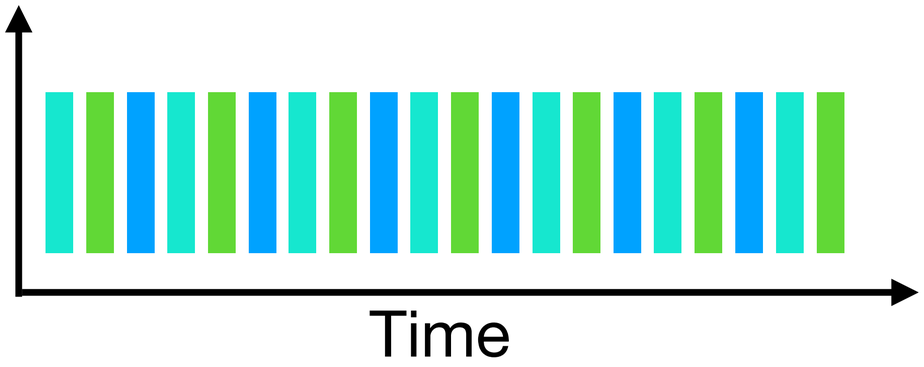
Space-Shared V.S. Time-Shared

The two kinds of scheduling

- **Space-Shared:** Each cloudlet gets exclusive access to the VM's resources until completion



- **Time-Shared:** Cloudlets share the VM's resources, and tasks are executed in time slices



Part II: Theory

Understanding the Problem

Intuition and mathematical formulation

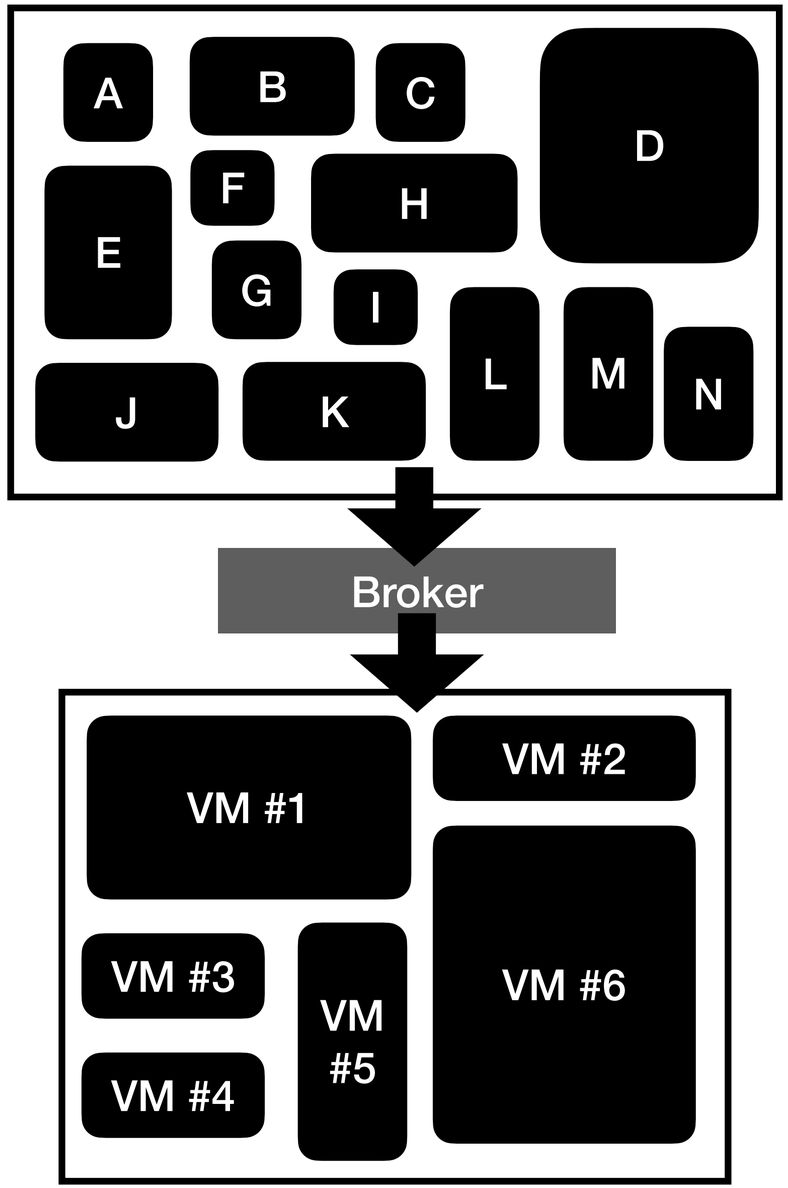
- We will consider the **makespan** (maximum time required to execute all cloudlets) as our main objective
- Cloudlet scheduling as a constrained optimization problem:
- Cloud task scheduling is an NP-hard problem

$$\text{Minimize } C_{\max} = \max_{j \in \{1,\ldots,M\}} \left(\sum_{i \in \{1,\ldots,N\}} x_{ij} \left(\frac{L_i}{P_j} + W_i \right) \right)$$

Subject to:

$$\sum_{j=1}^{M} x_{ij} = 1, \quad \forall i \in \{1, \ldots, N\}$$

$$\sum_{i=1}^{N} x_{ij} \cdot R_i \leq C_j, \quad \forall j \in \{1, \ldots, M\}$$

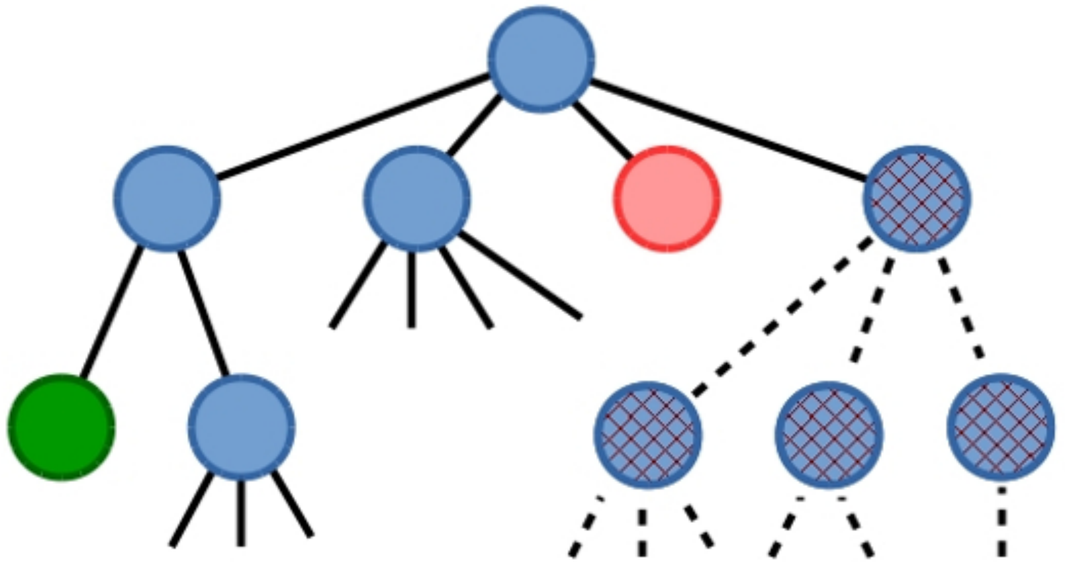


We will now examine 4 classes of scheduling algorithms

Exact Scheduling Algorithms

Optimal solutions, but at what cost?

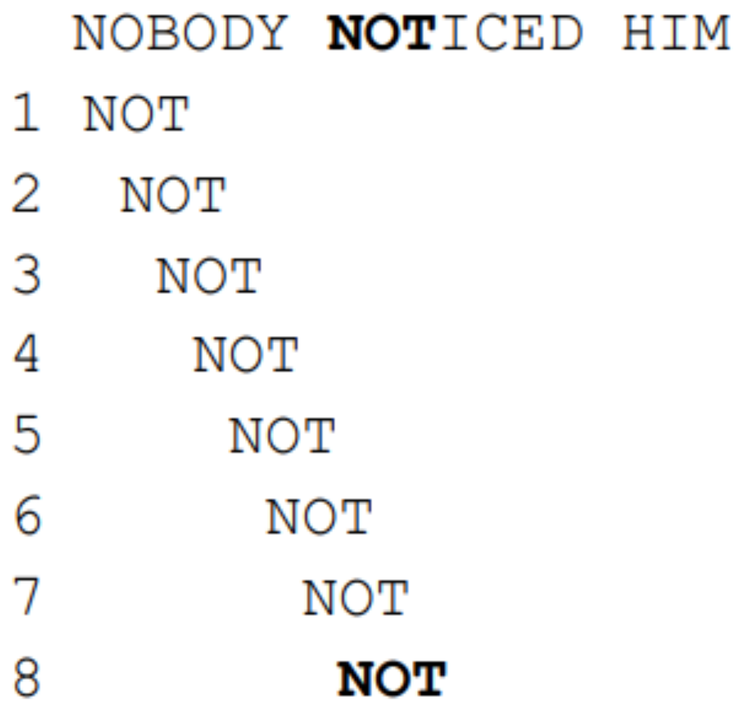
- The only class of algorithms that **guarantee** the best solutions
- Since cloudlet scheduling is NP-hard, making exact methods highly **impracticable**
- Different types exist, all with exponential complexity
  - Brute force
  - Dynamic programming
  - Branch and bound



Brute Forcing

The most basic optimal method

- Consists of three steps:
  - Consider all possible cloudlet assignments
  - Calculate the makespan for each solution
  - Return the best solution (lowest makespan)
- $O(M^N)$ complexity

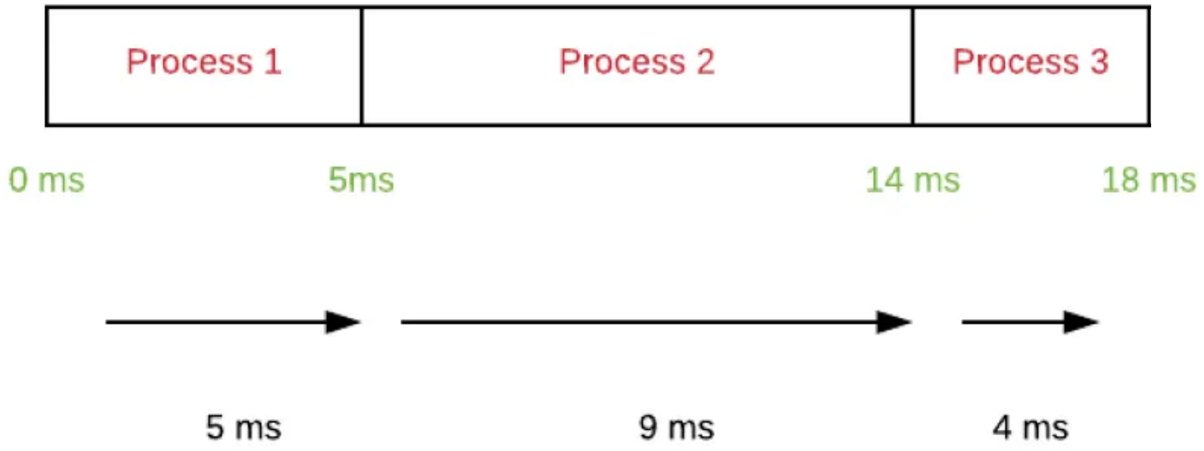


Brute force string matching

Heuristic Scheduling Algorithms

Trading optimality for efficiency

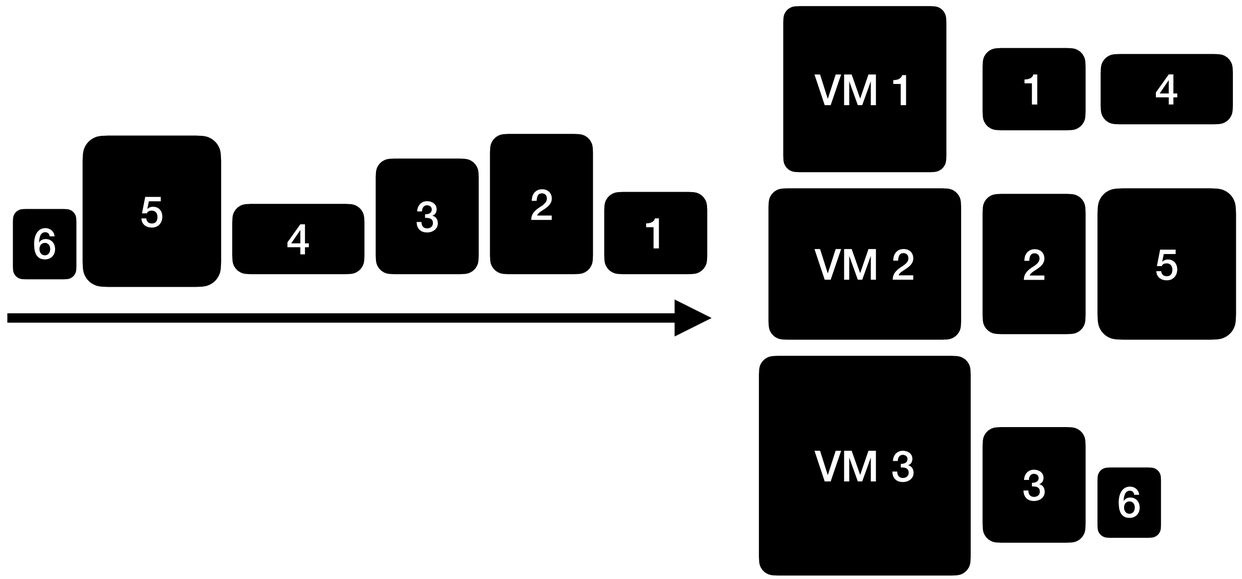
- “Use shortcuts or rules of thumb to find approximate solutions quickly”
- We turn to heuristic algorithms whenever classic methods are too slow
- Come in different shapes and sizes
  - FCFS
  - Round Robin
  - Shortest Job First (SJF)
  - Max-min & Min-min



Round Robin Scheduling

Splitting the load

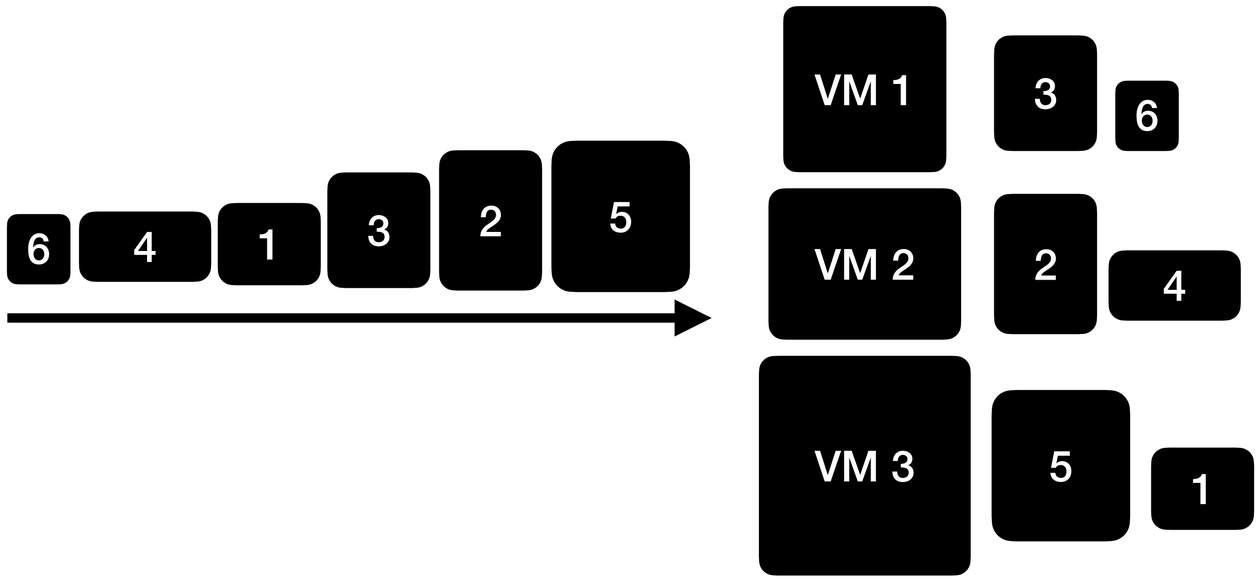
- Among the simplest heuristic methods
- Distributes the cloudlets among VMs in a circular fashion
- Despite its simplicity, usually has acceptable performance



Max-Min Scheduling

A more intelligent approach

- In RR, we pay no attention to the characteristics of the cloudlet or VM (MIPS, BW, …)
- How Max-Min works:
  - Select the task with the maximum execution time
  - Assign this task to the machine with the minimum completion time.
  - Repeat
- Requires more computation that RR (due to sorting)



Meta-Heuristic Scheduling Algorithms

One size fits all approaches

- “High-level and non problem-specific algorithms that find approximately solutions”
- A lot of them are inspired by nature (bio-inspired computing)
- Generally fall between exact and heuristic approaches in terms of computation and optimality
- Well over 100 distinct algorithms:
  - Genetic algorithm
  - Ant colony algorithm
  - wikipedia.org/wiki/Table_of_metaheuristics

Genetic Algorithm

Ant Colony Optimization

Artificial Bee Colony Optimization

Gravitational Search Algorithm

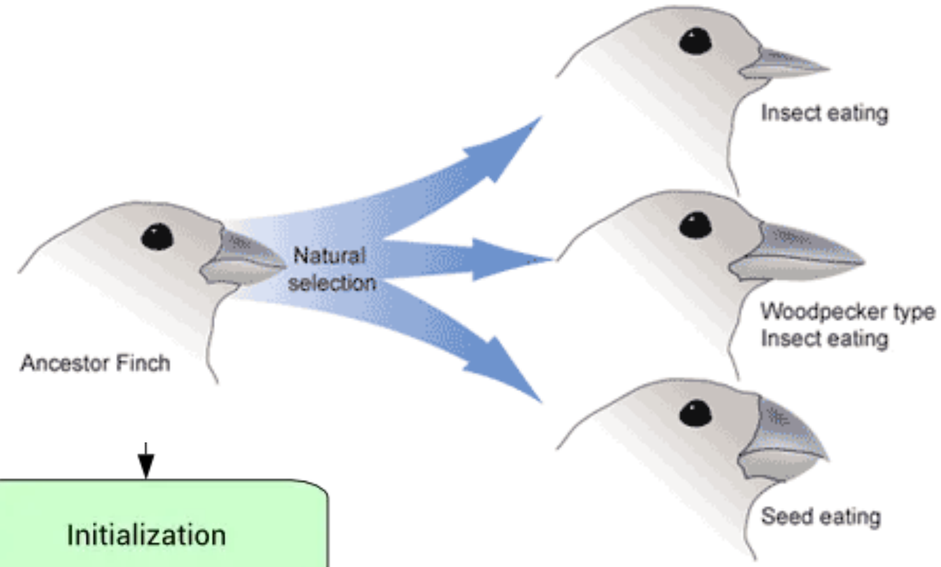
Firefly Algorithm

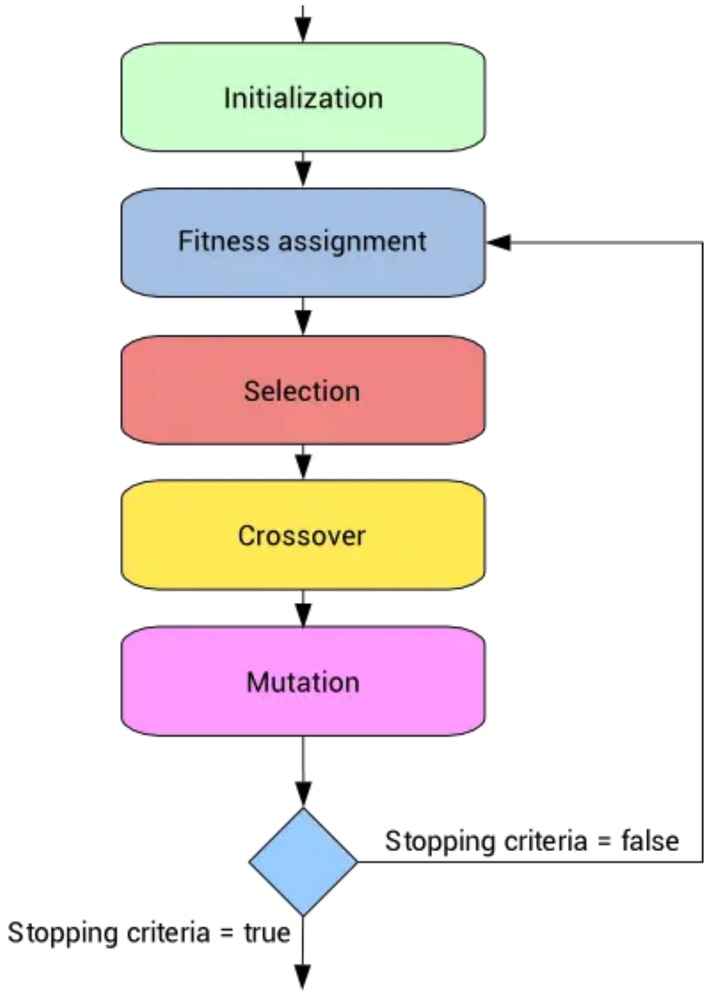
Fish Swarm Algorithm

Genetic Algorithms

Following in Darwin's footsteps

- Based on the theory of natural selection, the algorithm that has created **YOU**.
- Formulate each potential solution as an **individual**
- Evaluate the score of each individual using a **fitness function**
- **Select** the individuals with high levels of fitness
- Create offspring using the genes of the selected individuals (**crossover**) and **mutation**
- Randomly **mutate** the genes of newly created offspring

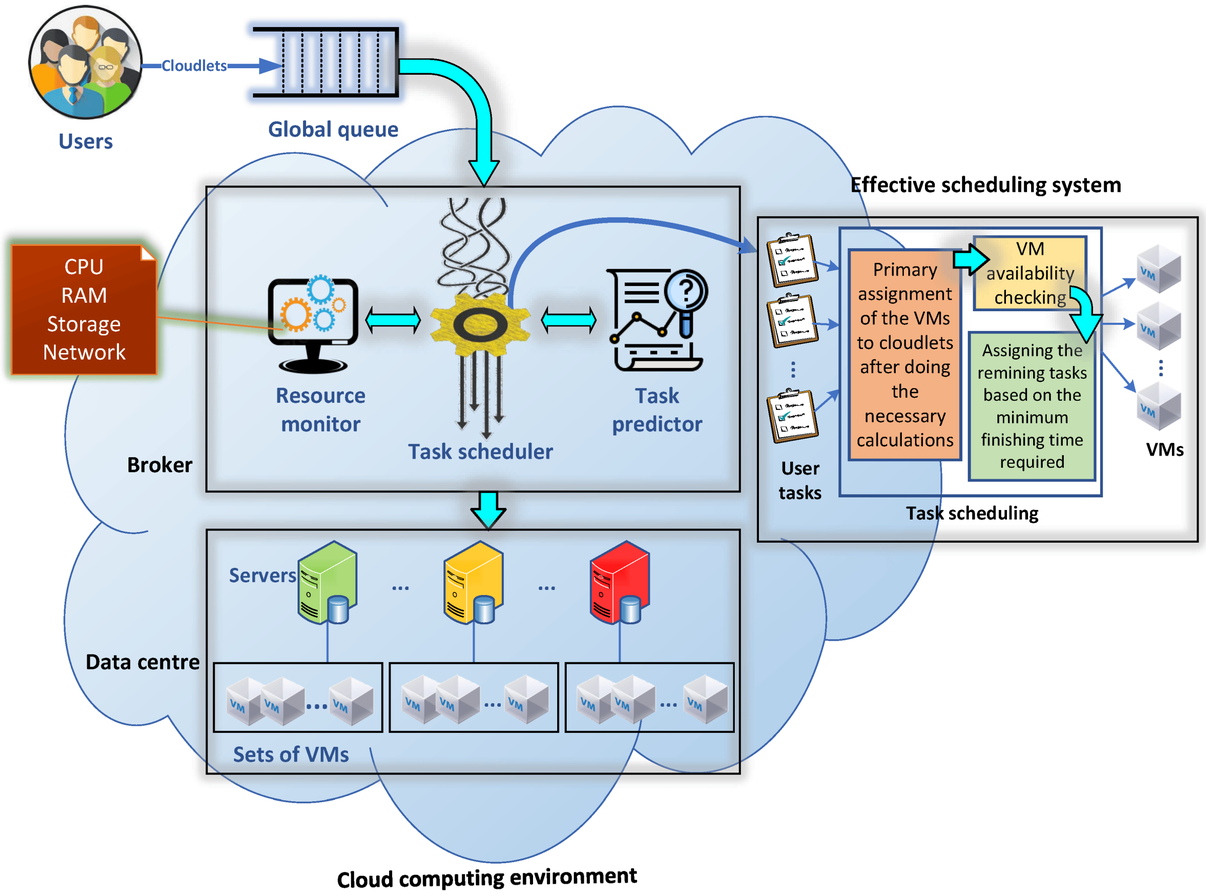




Learning-Based Scheduling

Rise of the artificial brains

- Based on Machine Learning techniques, generally using **deep neural networks**
- The hottest topic in CS and perhaps all of science
- Examples of ML for scheduling:
  - **Supervised Learning:** predict the future load of cloudiest
  - **Reinforcement Learning:** discover effective scheduling policies



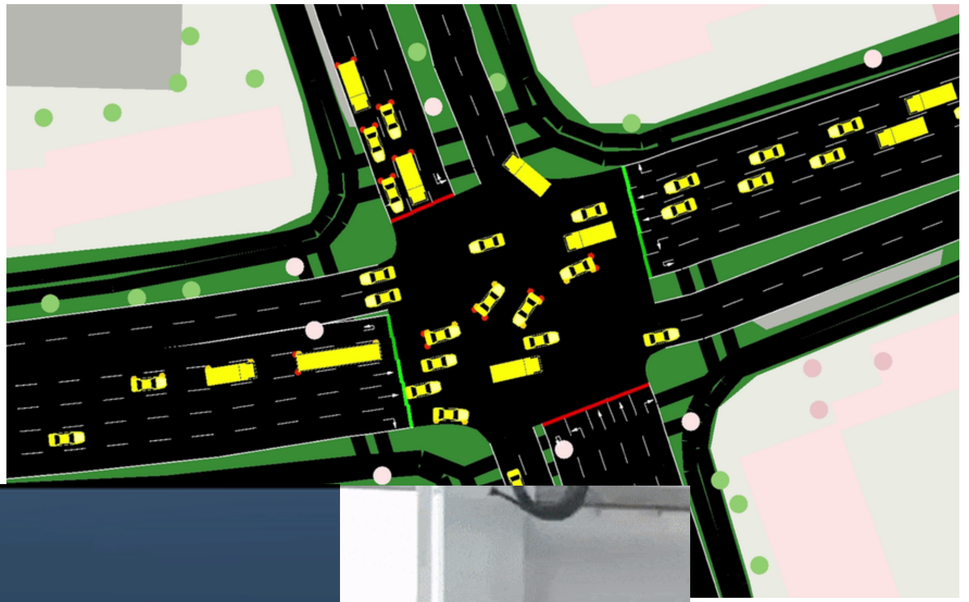
But what if we wanted to test a new scheduling algorithm?

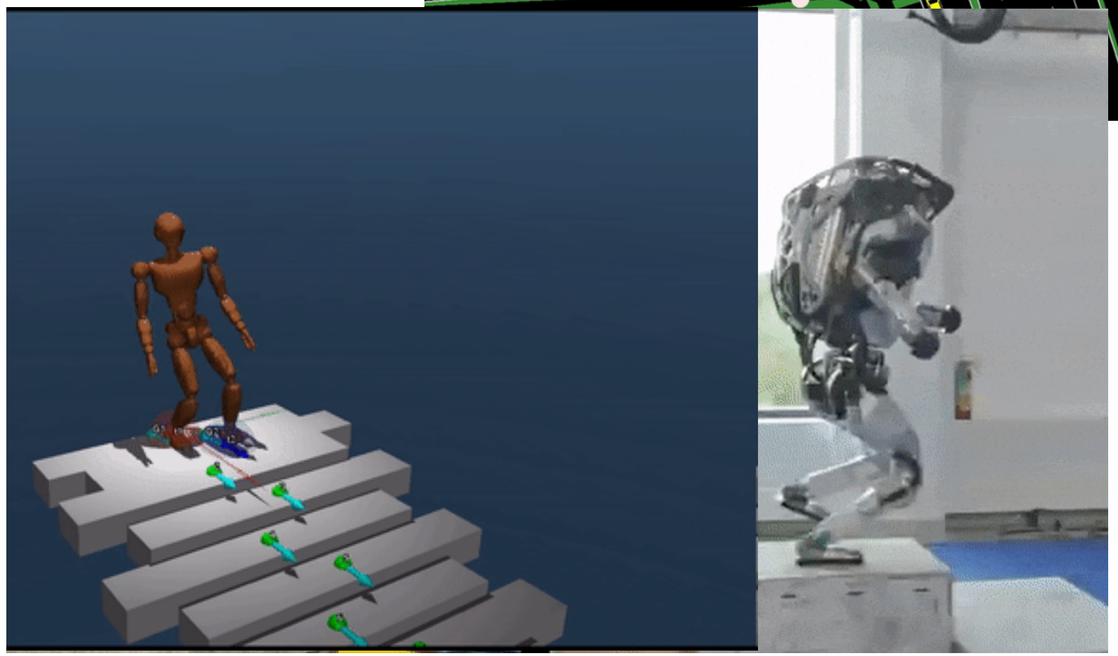
Part III: Simulation

Computer Simulators

The best testbed for verifying new ideas

- Novel ideas require multiple iterations of improvement to work
- Testing these new ideas in the real-world consumes times, money, or are straight out dangerous
- The more accurate the simulation, the better





CloudSim

The premier cloud simulator

- From the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne
- Completely open-sourced
- Written in pure Java (OOP)
- Widely used in academia and industry

CloudSim: a **toolkit** for **modeling** and **simulation** of **cloud computing environments** and **evaluation** of **resource provisioning algorithms**

RN Calheiros, R Ranjan, A Beloglazov… - Software: Practice …, 2011 - Wiley Online Library

…) system **simulators** 8-10 offer the **environment** that can be directly used … **modeling Cloud computing environments**, we present **CloudSim**: a new, generalized, and extensible **simulation** …

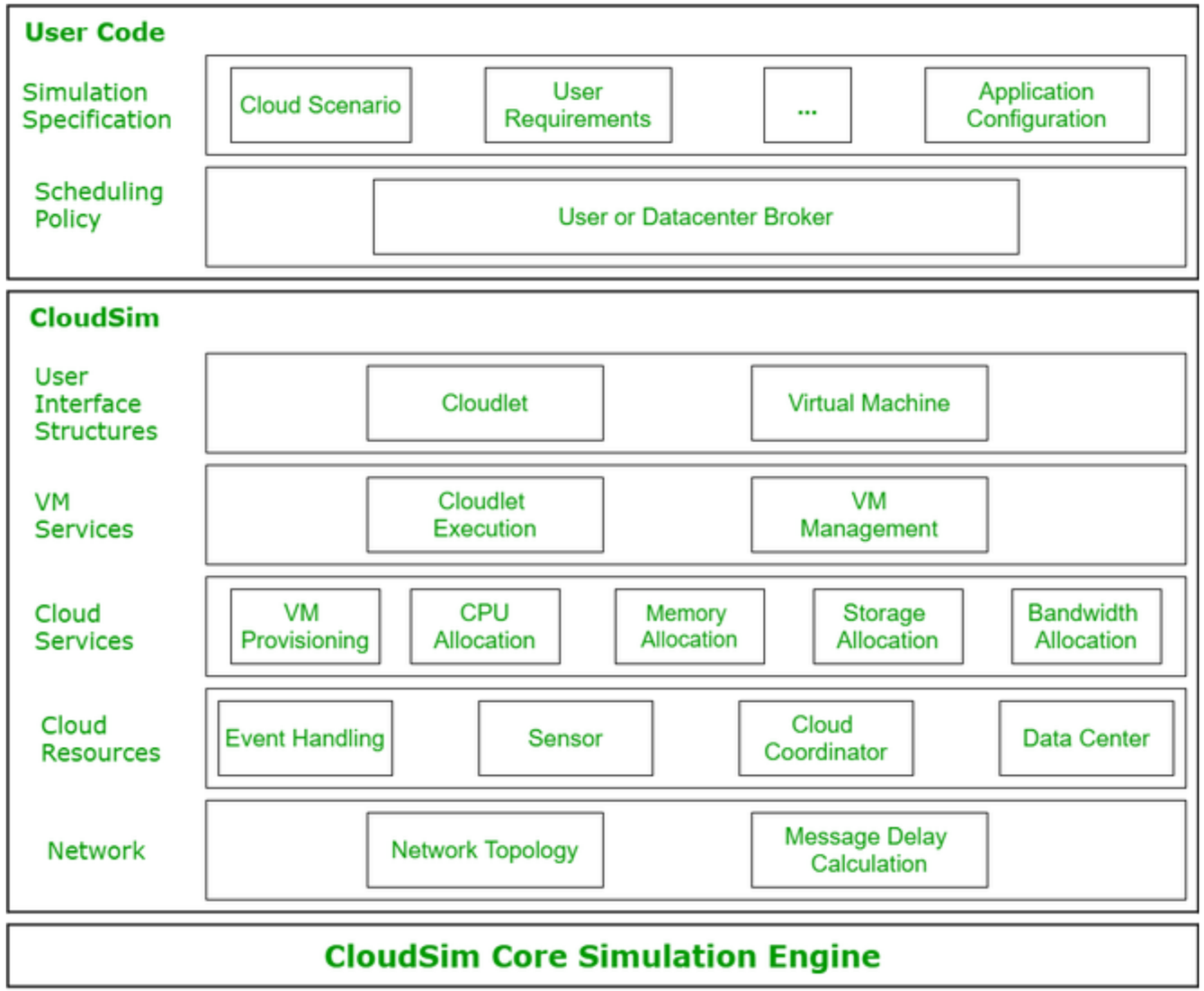
☆ Save Cite Cited by 6459 Related articles All 23 versions

Prof. Rajkumar Buyya

Director of CLOUDS

CloudSim's Architecture

- Layered architecture, changing the upper layers is more straightforward
- Average users work in the "User Code" or the "CloudSim" layer
- Changing the core engine code requires a deeper understanding of the software



Installing CloudSim

From installation to hello world

- The official installation guide: cloudsimtutorials.online/cloudsim-setup-using-eclipse/
- The four things needed:
  - Java Development Kit (JDK)
  - A Java IDE (preferably Eclipse)
  - CloudSim sourcecode (https://github.com/Cloudslab/cloudsim)
  - Apache commons math package

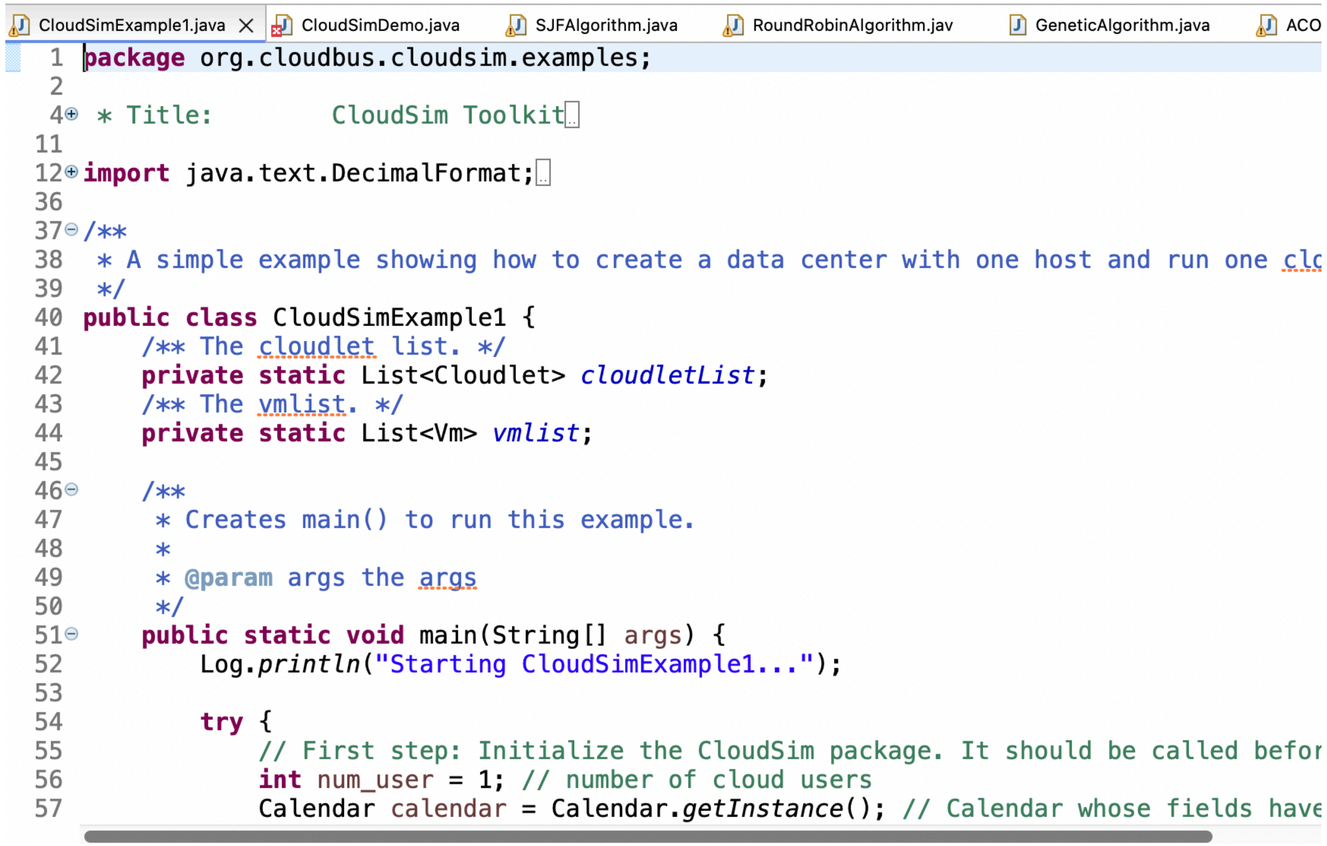
Essential CloudSim Classes

- **Cloudlet**
  - ID
  - Length
  - pesNumber
  - cloudletFileSize
  - cloudletOutputSize
- **VM**
  - ID
  - MIPS
  - pesNumber
  - Bw
  - Size
  - CloudletScheduler
- **Host**
  - …
- **Broker**
  - …
- **DataCenter**
  - …

CloudSim Example 1

Dipping our toes into the water

- Perhaps the simplest scenario one can create with CloudSim
- Part of a series of 9 example codes provided by CloudSim
- Creates a single cloudlet and executes it on a single VM
- Now, we will see it in action!



CloudSim Example 9

A deeper glimpse into CloudSim

- Showcases the differences between time-shared and space-shared execution
- We skip over Examples 2-8 due to time
- Overall, very similar in structure to Example 1
- Now, we will see it in action!

CloudSimExample9.java

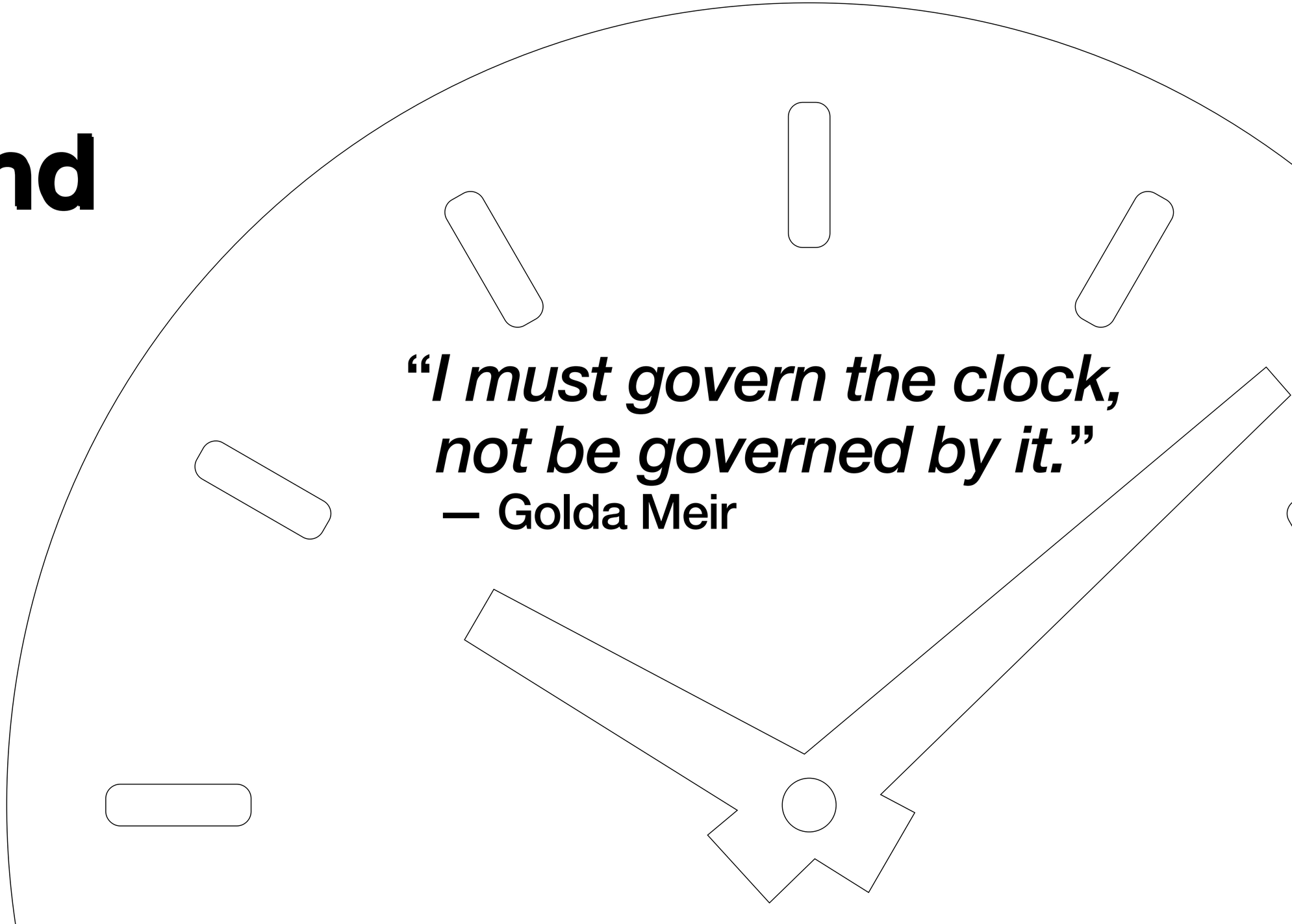
```
1  package org.cloudbus.cloudsim.examples;
2
4  * Title:        CloudSim Toolkit
11
12 import org.cloudbus.cloudsim.*;
20
21 /**
22  * A simple example showing the 2 cloudlet scheduling models: time-shared and space-shared.
23  *
24  * @author Remo Andreoli
25  */
26 public class CloudSimExample9 {
27     /** The cloudlet list. */
28     private static List<Cloudlet> cloudletList;
29     /** The vmlist. */
30     private static List<Vm> vmlist;
31
32     /**
33      * Creates main() to run this example.
34      *
35      * @param args the args
36      */
37     public static void main(String[] args) {
38         Log.printLine("Starting CloudSimExample9...");
39
40         try {
41             // First step: Initialize the CloudSim package. It should be called before creat
42             int num_user = 1; // number of cloud users
43             Calendar calendar = Calendar.getInstance(); // Calendar whose fields have been
44             boolean trace_flag = false; // trace events
45
46             CloudSim.init(num_user, calendar, trace_flag);
47
48             // Second step: Create Datacenters
49             // Datacenters are the resource providers in CloudSim. We need at
```

Custom Scheduling

Implementation and comparison of algorithms

- We have implemented the following algorithms in CloudSim:
  - Brute Force
  - Round Robin
  - Max-Min
  - Genetic Algorithm
  - Ant Colony Optimization
- We will now provide a detailed explanation

The End



“I must govern the clock, not be governed by it.”

— Golda Meir