Data Integration and Large Scale Analysis

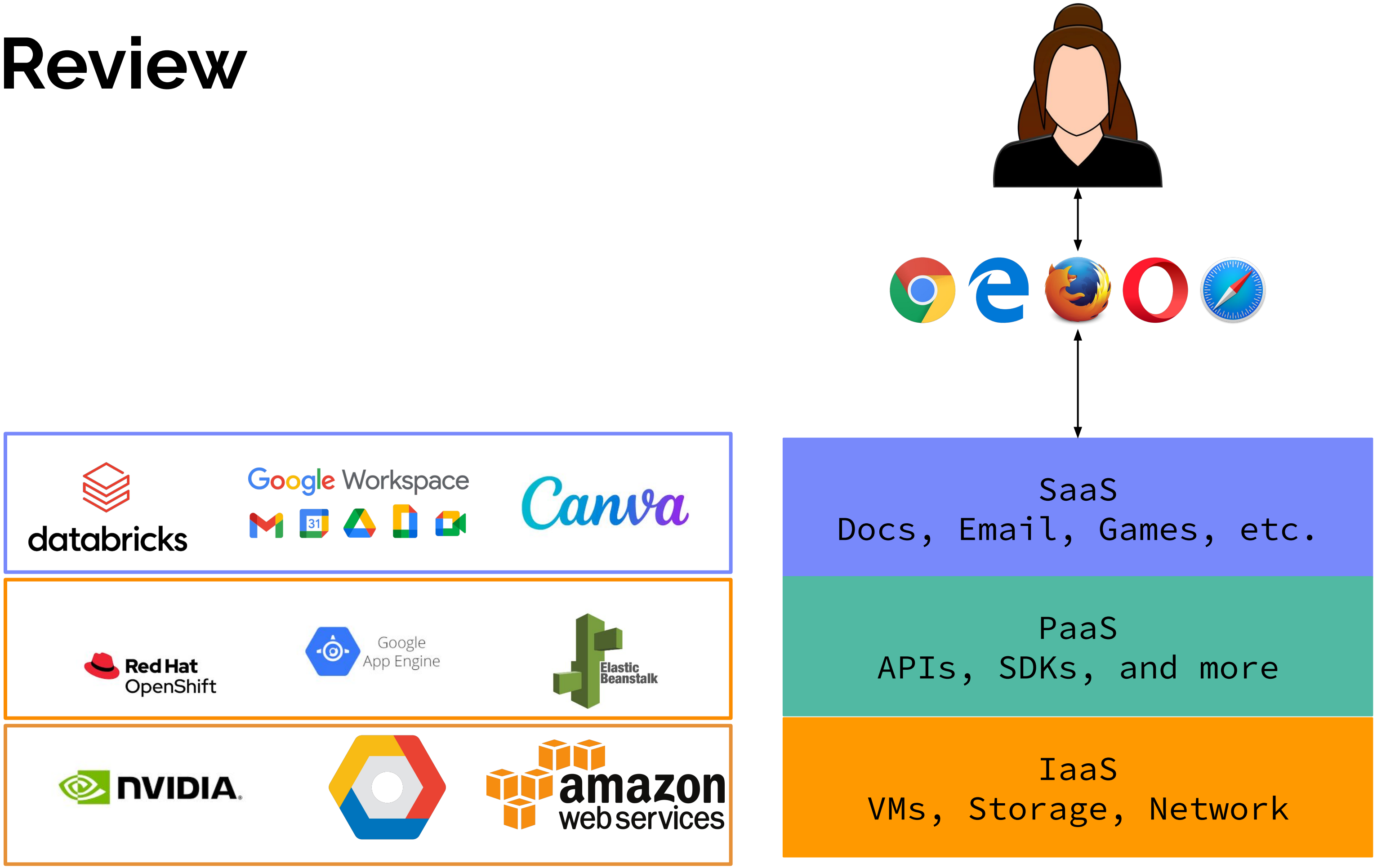
09- Cloud Resource Management and Scheduling

Lucas Iacono. PhD. - 2024

Learning Objectives

- Understand the importance of resource management and **scheduling** in cloud computing.
- Learn about common scheduling algorithms and their applications.
- Explore real-world use cases and research trends in cloud resource scheduling.
- Gain practical competences in scheduling algorithms.

Review


databricks
Google Workspace
Canva
Red Hat OpenShift
Google App Engine
Elastic Beanstalk
NVIDIA
amazon web services
SaaS
Docs, Email, Games, etc.
PaaS
APIs, SDKs, and more
IaaS
VMs, Storage, Network

Agenda

- Motivation and Terminology
- Cloud Computing Scheduling
- Activity

Motivation and Terminology

Motivation and Terminology

- How I can feed my hungry app with computing and storage resources?

What is Cloud Resource Management?

(Easy) Definition:

The process of efficiently allocating **computational, storage, and network resources** to meet the needs of applications and users in cloud environments.

Key Factors:

- **Availability:** Are resources ready for use?
- **Efficiency:** Are resources optimally utilized?
- **Cost:** Are costs minimized while meeting objectives?

Example:

Netflix manages server capacity during peak hours to support millions of users streaming simultaneously.

Fortnite live shows

Types of Cloud Resources

Computational Resources:

- CPUs, GPUs
- Example: AWS EC2 instances /NVDIA GPUs .

Storage Resources:

- Distributed file systems, databases.
- Example: Amazon S3 or Google Cloud Storage.

Network Resources:

- **Bandwidth, load balancing, connectivity.**
- **Example:** Cloudflare's CDN for global content delivery.

What is Scheduling?

Definition:

Scheduling involves assigning tasks to available resources efficiently, based on predefined criteria such as priority, execution time, or costs.

Why is it Important?

- Enhances **QoS.**
- Reduces **costs** through better resource utilization.
- Prevents **overloading or underutilization** of resources.

Example:

- **Training an AI model on GPUs is scheduled at night to minimize cost.**

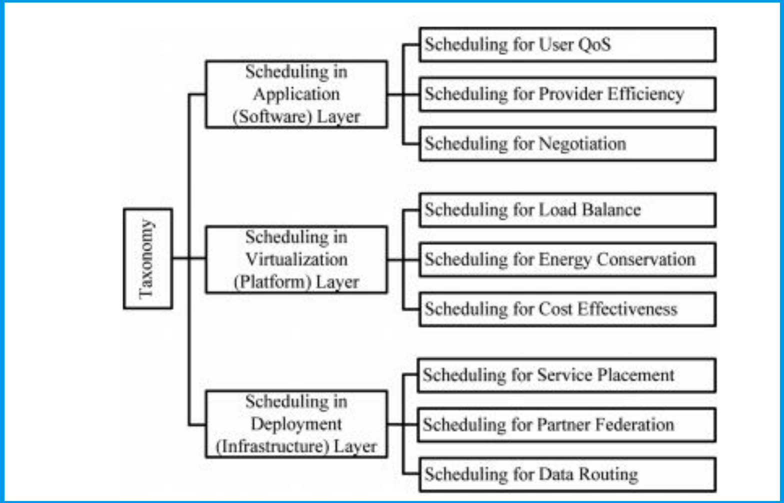
What is Scheduling?

Taxonomy of Cloud Resource Scheduling


Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches

ZHI-HUI ZHAN, XIAO-FANG LIU, YUE-JIAO GONG, and JUN ZHANG,
HENRY SHU-HUNG CHUNG,
YUN LI

Zhan, Z. H., Liu, X. F., Gong, Y. J., Zhang, J., Chung, H. S. H., & Li, Y. (2015). Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Computing Surveys (CSUR), 47(4), 1-33.

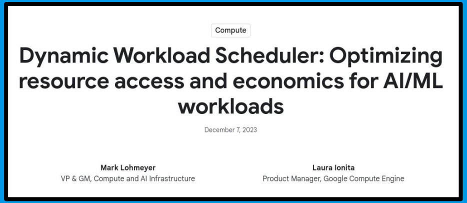


Cloud Computing Scheduling

Key Factors in Scheduling

- **Task Priority:**
  - Critical tasks are prioritized.
  - **Example:** Emergency vehicle routing.
- **Execution Time:**
  - Tasks with shorter times may be prioritized to optimize throughput.
  - **Example:** Obstacle detection.
- **Load Balancing:**
  - Ensures resources are equally utilized.
  - **Example:** AWS regions balancing traffic loads to compute traffic planning.
- **Energy Efficiency:**
  - Optimizes to reduce power consumption.
  - Example: Shutting down idle servers during low usage periods (few vehicles at night).

Challenges in Scheduling


Compute

Dynamic Workload Scheduler: Optimizing resource access and economics for AI/ML workloads

December 7, 2023

Mark Lohmeyer
VP & GM, Compute and AI Infrastructure

Laura Ionita
Product Manager, Google Compute Engine

Lohmeyer, M., Ionita, L., (2023). Dynamic Workload Scheduler: Optimizing resource access and economics for AI/ML workloads. Google Cloud Blog.

Energy Optimization:

- Balancing performance with reduced energy consumption.

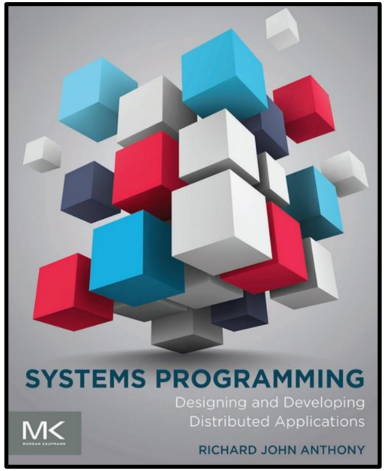
Scheduling in Hybrid Environments:

- Example: Combining edge devices and cloud servers for IoT applications.

AI-Driven Scheduling:

- Using Machine Learning to predict demand and optimize scheduling.
- Example: Google Cloud's Dynamic Workload Scheduler

Common Scheduling Algorithms



Anthony, R. (2015). *Systems programming: designing and developing distributed applications*. Morgan Kaufmann.

First-Come, First-Served (FCFS):

Assigns tasks in the **order they arrive.**

- **Advantage:** Simple to implement.
- **Disadvantage:** Inefficient if long tasks arrive first.
- **Example**
  a. Tasks: **T1 (2s)**, **T2 (4s)**, **T3 (1s)**
  b. Execution order: **T1** → **T2** → **T3**
- **Total time: 2 + 4 + 1 = 7 seconds.**

```
Procedure FCFS_Scheduling(tasks):
    Initialize total_time = 0
    For each task in tasks:
        Print "Executing task:", task.id, "Execution
time:", task.execution_time
        total_time = total_time + task.execution_time
    End For
    Print "Total execution time:", total_time
End Procedure
```

1. **Receives a list of tasks with their execution times.**
2. **Iterates through each task and sums its execution time to the total.**
3. **Prints the total execution time.**

Common Scheduling Algorithms

Round **Robin:**

- Allocates a fixed time slice (quantum) to each task in a cyclic order.
- **Example:**
- **Tasks:** **T1 (2s)**, **T2 (4s)**, **T3 (1s)**
- **Quantum:** **1s**
- **Execution order:**
    a. Round 1: **T1** (**1s**), **T2** (**1s**), **T3** (**1s**)
    b. Round 2: **T1** (**1s**), **T2** (**1s**)
    c. Round 3: **T2** (**1s**)
    d. Round 4: **T2** (**1s**)
- **Total time: 7 seconds.**

```
Procedure RoundRobin_Scheduling(tasks, quantum):
    Initialize total_time = 0
    While tasks is not empty:
        For each task in tasks:
            If task.execution_time > quantum:
                Print "Executing:", task.id, "for", quantum, "units"
                task.execution_time = task.execution_time - quantum
                total_time = total_time + quantum
            Else:
                Print "Completing task:", task.id, "Remaining time:",
task.execution_time
                total_time = total_time + task.execution_time
                Remove task from tasks
            End If
        End For
    End While
    Print "Total execution time:", total_time
End Procedure
```

1. **Each task executes for a maximum of quantum time units.**
2. **If a task is not completed, its remaining time is reduced, and it is rescheduled.**
3. **Repeats until all tasks are completed.**

Common Scheduling Algorithms

Min-Min and Max-Min:

- **Min-Min:** Assigns **shortest** tasks **first**.
- **Max-Min:** Assigns **longest** tasks **first**.
- **Tasks: T1 (2s), T2 (4s), T3 (1s)**
- **Resources: S1, S2**
- **Assignment order (Min-Min):**
    a. **T3 → S1**
    b. **T1 → S2**
    c. **T2 → S1**
- **Assignment order (Max-Min):**
    a. **T2 → S1**
    b. **T1 → S2**
    c. **T3 → S1**

```
Procedure MinMin_Scheduling(tasks, resources):
    While tasks is not empty:
        Find task_min = Task with the shortest execution_time
        Assign task_min to the least loaded resource
        Remove task_min from tasks
        Print "Assigning task:", task_min.id, "to the least loaded
resource"
    End While
End Procedure
```

- **Finds the task with the shortest execution time in each iteration.**
- **Assigns the task to the least loaded resource.**
- **Repeats until no tasks are left.**

Common Scheduling Algorithms

Load Balancing:

- **Distributes tasks equally across all available resources.**
- **Tasks: T1 (2s), T2 (4s), T3 (1s)**
- **Resources: S1, S2**
- **Result:**
    a. **0 → S1, 0 →S2**
    b. **T1 (2s) → S1, 0 → S2 (Total load: 2)**
    c. **T1 (2s) → S1, T2 (4s) → S2 (Total load: 6)**
    d. **T3 (1s) + T1 (2s)→ S1, T2 (4s) →S2 (Total load: 7)**
    e. **S1 = 3s , S2 = 4s**

```
Procedure LoadBalancing_Scheduling(tasks, resources):
    Initialize resource_load = [0 for each resource]
    For each task in tasks:
        Find least_loaded_resource = Resource with minimum load
        Assign task to least_loaded_resource
        Update resource_load for least_loaded_resource
        Print "Task:", task.id, "assigned to resource:",
least_loaded_resource
    End For
End Procedure
```

1. **Initializes the load for each resource to 0.**
2. **Assigns each task to the least loaded resource.**
3. **Updates the load for the corresponding resource.**

Algorithm	Advantage	Disadvantage	Ideal Use Case
FCFS	Simple and easy to implement	Inefficient with long tasks first	Light and homogeneous workloads (e.g. Max and Min temp detection in IoT time series)
Round Robin	Fair, avoids resource blocking	Does not optimize execution times	Interactive processing (Human-Machine interaction)
Min-Min	Optimizes execution times	Requires global analysis	Large workloads with varied tasks (HPC, short tasks and makespan minimization)
Load Balancing	Balances resource usage	Ignores task priority or execution time	Real-time scenarios (Web servers load balance to manage users' requests).

Advanced Scheduling Algorithms

Heuristic-based

- Genetic Algorithms
- Particle Swarm Optimization
- Others (simulated annealing, etc.).

Used for **complex, large-scale** systems.

What are Metaheuristics?

Definition:

- **Metaheuristics** are high-level optimization algorithms designed to find **approximate solutions** for complex problems that are h**ard to solve using traditional methods**.

Key Features:

- **General Framework:** Not problem-specific.
- **Exploration and Exploitation:** Search new domains (exploration) and refining existing solutions (exploitation).
- **Scalability:** Works well with large, complex problems.

Examples:

- Genetic Algorithms (GA)
- Particle Swarm Optimization (PSO)
- Simulated Annealing

Metaheuristics for Scheduling?

Complexity of Scheduling:

- Traditional algorithms (e.g., FCFS, Round Robin) struggle with **high-dimensional** or **dynamic scheduling problems**.

Dynamic Environments:

- **Real-time** adjustments based on **changing workloads** and r**esource availability.**

Multi-Objective Optimization:

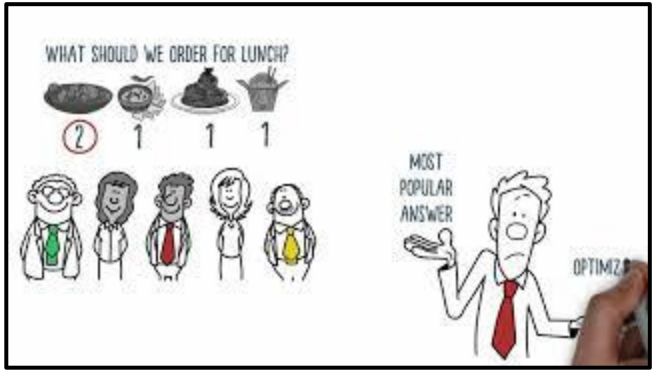
- **Balances** conflicting goals like **cost, energy consumption, and performance**.

Example:

- Allocating tasks in a hybrid cloud environment where some tasks prioritize speed while others prioritize cost efficiency.

Metaheuristics

Particle Swarm



Metaheuristics

Particle Swarm Optimization: Frost Prediction

- **Application**: Frost Prediction Applications
- **Challenge:** Efficient scheduling of CPU-intensive tasks in federated clouds, minimizing makespan (**execution time**) and **monetary cost**.
- **Federated Clouds:** Utilized for distributed computing across geographically dispersed datacenters.

Journal of Network and Systems Management (2019) 27:688–729
https://doi.org/10.1007/s10922-018-9481-0

A Bio-inspired Datacenter Selection Scheduler for Federated Clouds and Its Application to Frost Prediction

Elina Pacini[1,2] · Lucas Iacono[1,2] · Cristian Mateos[3] · Carlos García Garino[1]

Received: 14 August 2017 / Revised: 1 November 2018 / Accepted: 8 November 2018 /
Published online: 22 November 2018

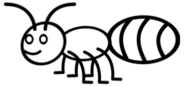
Abstract
Frost is an agro-meteorological event which causes both damage in crops and important economic losses, therefore frost prediction applications (FPA) are very important to help farmers to mitigate possible damages. FPA involves the execution of many CPU-intensive jobs. This work focuses on efficiently running FPAs in paid federated Clouds, where custom virtual machines (VM) are launched in appropriate resources belonging to different providers. The goal of this work is to minimize both the makespan and monetary cost. We follow a federated Cloud model where scheduling is performed at three levels. First, at the *broker level*, a datacenter is selected taking into account certain criteria established by the user, such as lower costs or lower latencies. Second, at the *infrastructure level*, a specialized scheduler is responsible for mapping VMs to datacenter hosts. Finally, at the *VM level*, jobs are assigned for execution into the preallocated VMs. Our proposal mainly contributes to implementing bio-inspired strategies at two levels. Specifically, two broker-level schedulers based on Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), which aim to select the datacenters taking into account the network latencies, monetary cost and the availability of computational resources in datacenters, are implemented. Then, VMs are allocated in the physical machines of that datacenter by another intra-datacenter scheduler also based on ACO and PSO. Performed experiments show that our bio-inspired scheduler succeed in reducing both the makespan and the monetary cost with average gains of around 50% compared to genetic algorithms.

Keywords Scientific computing · Frost prediction applications · Cloud computing · Scheduling · Ant colony optimization · Particle Swarm optimization · Genetic algorithms

Pacini, E., Iacono, L., Mateos, C., & García Garino, C. (2019). A bio-inspired datacenter selection scheduler for federated clouds and its application to frost prediction. Journal of Network and Systems Management, 27(3), 688-729.

Metaheuristics: Particle Swarm Optimization

Key Aspects:

- Two schedulers **PSO** and **ACO** 
- Implemented at the **broker** (**datacenter selection**) and **IaaS (VM allocation)** levels

Multi-objective Optimization:

- Balanced trade-offs between makespan, monetary cost, and resource availability.
- Included considerations for **network latencies** and **datacenter capacities**.

Journal of Network and Systems Management (2019) 27:688–729
https://doi.org/10.1007/s10922-018-9481-0

CrossMark

A Bio-inspired Datacenter Selection Scheduler for Federated Clouds and Its Application to Frost Prediction

Elina Pacini[1,2] · Lucas Iacono[1,2] · Cristian Mateos[3] · Carlos García Garino[1]

Received: 14 August 2017 / Revised: 1 November 2018 / Accepted: 8 November 2018 /
Published online: 22 November 2018

Abstract
Frost is an agro-meteorological event which causes both damage in crops and important economic losses, therefore frost prediction applications (FPA) are very important to help farmers to mitigate possible damages. FPA involves the execution of many CPU-intensive jobs. This work focuses on efficiently running FPAs in paid federated Clouds, where custom virtual machines (VM) are launched in appropriate resources belonging to different providers. The goal of this work is to minimize both the makespan and monetary cost. We follow a federated Cloud model where scheduling is performed at three levels. First, at the *broker level*, a datacenter is selected taking into account certain criteria established by the user, such as lower costs or lower latencies. Second, at the *infrastructure level*, a specialized scheduler is responsible for mapping VMs to datacenter hosts. Finally, at the *VM level*, jobs are assigned for execution into the preallocated VMs. Our proposal mainly contributes to implementing bio-inspired strategies at two levels. Specifically, two broker-level schedulers based on Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), which aim to select the datacenters taking into account the network latencies, monetary cost and the availability of computational resources in datacenters, are implemented. Then, VMs are allocated in the physical machines of that datacenter by another intra-datacenter scheduler also based on ACO and PSO. Performed experiments show that our bio-inspired scheduler succeed in reducing both the makespan and the monetary cost with average gains of around 50% compared to genetic algorithms.

Keywords Scientific computing · Frost prediction applications · Cloud computing · Scheduling · Ant colony optimization · Particle Swarm optimization · Genetic algorithms

Pacini, E., Iacono, L., Mateos, C., & García Garino, C. (2019). A bio-inspired datacenter selection scheduler for federated clouds and its application to frost prediction. Journal of Network and Systems Management, 27(3), 688-729.


Journal of Network and Systems Management (2019) 27:688–729
https://doi.org/10.1007/s10922-018-9481-0

CrossMark

A Bio-inspired Datacenter Selection Scheduler for Federated Clouds and Its Application to Frost Prediction

Elina Pacini[1,2] · Lucas Iacono[1,2] · Cristian Mateos[3] · Carlos García Garino[1]

Received: 14 August 2017 / Revised: 1 November 2018 / Accepted: 8 November 2018 /
Published online: 22 November 2018

Abstract
Frost is an agro-meteorological event which causes both damage in crops and important economic losses, therefore frost prediction applications (FPA) are very important to help farmers to mitigate possible damages. FPA involves the execution of many CPU-intensive jobs. This work focuses on efficiently running FPAs in paid federated Clouds, where custom virtual machines (VM) are launched in appropriate resources belonging to different providers. The goal of this work is to minimize both the makespan and monetary cost. We follow a federated Cloud model where scheduling is performed at three levels. First, at the *broker level*, a datacenter is selected taking into account certain criteria established by the user, such as lower costs or lower latencies. Second, at the *infrastructure level*, a specialized scheduler is responsible for mapping VMs to datacenter hosts. Finally, at the *VM level*, jobs are assigned for execution into the preallocated VMs. Our proposal mainly contributes to implementing bio-inspired strategies at two levels. Specifically, two broker-level schedulers based on Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), which aim to select the datacenters taking into account the network latencies, monetary cost and the availability of computational resources in datacenters, are implemented. Then, VMs are allocated in the physical machines of that datacenter by another intra-datacenter scheduler also based on ACO and PSO. Performed experiments show that our bio-inspired scheduler succeed in reducing both the makespan and the monetary cost with average gains of around 50% compared to genetic algorithms.

Keywords Scientific computing · Frost prediction applications · Cloud computing · Scheduling · Ant colony optimization · Particle Swarm optimization · Genetic algorithms

Pacini, E., Iacono, L., Mateos, C., & García Garino, C. (2019). A bio-inspired datacenter selection scheduler for federated clouds and its application to frost prediction. Journal of Network and Systems Management, 27(3), 688-729.

Metaheuristics: Particle Swarm Optimization

Broker-Level Scheduler:

- **Selects** datacenters considering communication latency and monetary cost using PSO and ACO.
- Weighs **monetary cost** (e.g., VM pricing) and **latency** with adjustable parameters.

Infrastructure-Level Scheduler:

- **Allocates** VMs to datacenter hosts.
- **Ensures** efficient use of physical resources to minimize costs and execution delays.

VM-Level Scheduler:

- FCFS-based job scheduling within preallocated VMs.

Metaheuristics: Particle Swarm Optimization

Experimental Validation:

- Simulated frost applications with real-world frost prediction data using CloudSim.
- Achieved **50% reduction in makespan and monetary costs** compared to traditional Genetic Algorithms (GAs).

Advantages of PSO:

- **Faster convergence** and **adaptability** to dynamic cloud environments.
- **Effective** in **balancing load** among heterogeneous datacenters.


Journal of Network and Systems Management (2019) 27:688–729
https://doi.org/10.1007/s10922-018-9481-0

CrossMark

A Bio-inspired Datacenter Selection Scheduler for Federated Clouds and Its Application to Frost Prediction

Elina Pacini[1,2] · Lucas Iacono[1,2] · Cristian Mateos[3] · Carlos García Garino[1]

Received: 14 August 2017 / Revised: 1 November 2018 / Accepted: 8 November 2018 / Published online: 22 November 2018

Abstract
Frost is an agro-meteorological event which causes both damage in crops and important economic losses, therefore frost prediction applications (FPA) are very important to help farmers to mitigate possible damages. FPA involves the execution of many CPU-intensive jobs. This work focuses on efficiently running FPAs in paid federated Clouds, where custom virtual machines (VM) are launched in appropriate resources belonging to different providers. The goal of this work is to minimize both the makespan and monetary cost. We follow a federated Cloud model where scheduling is performed at three levels. First, at the *broker level*, a datacenter is selected taking into account certain criteria established by the user, such as lower costs or lower latencies. Second, at the *infrastructure level*, a specialized scheduler is responsible for mapping VMs to datacenter hosts. Finally, at the *VM level*, jobs are assigned for execution into the preallocated VMs. Our proposal mainly contributes to implementing bio-inspired strategies at two levels. Specifically, two broker-level schedulers based on Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), which aim to select the datacenters taking into account the network latencies, monetary cost and the availability of computational resources in datacenters, are implemented. Then, VMs are allocated in the physical machines of that datacenter by another intra-datacenter scheduler also based on ACO and PSO. Performed experiments show that our bio-inspired scheduler succeed in reducing both the makespan and the monetary cost with average gains of around 50% compared to genetic algorithms.

Keywords Scientific computing · Frost prediction applications · Cloud computing · Scheduling · Ant colony optimization · Particle Swarm optimization · Genetic algorithms

Pacini, E., Iacono, L., Mateos, C., & García Garino, C. (2019). A bio-inspired datacenter selection scheduler for federated clouds and its application to frost prediction. Journal of Network and Systems Management, 27(3), 688-729.

```
Procedure PSO(tasks, num_particles, iterations):
    Initialize particles with random positions (schedules) and
velocities
    For iteration in 1 to iterations:
        For each particle:
            Evaluate fitness of the particle's position
            Update personal best and global best positions
            Adjust velocity based on personal and global best
            Update particle's position
        End For
    End For
    Return global best schedule
End Procedure
```

Key Components:

- **Particle: A potential schedule.**
- **Velocity: How a particle adjusts its solution.**
- **Global Best: The best solution found so far.**

Other Metaheuristics

Genetic Algorithms (GA):

- Inspired by **natural selection.**

Process:

- **Selection:** Choose the best solutions.
- **Crossover:** Combine solutions to create new ones.
- **Mutation:** Introduce randomness for diversity.

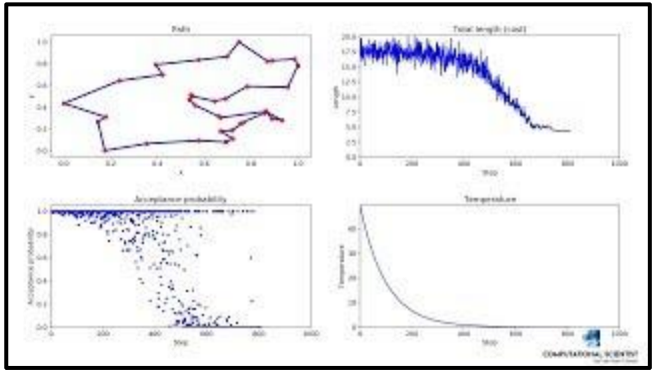
Use Case: Optimizing resource allocation in data centers.

```
Procedure GeneticAlgorithm(tasks, population_size, generations):
    Initialize population with random schedules
    For generation in 1 to generations:
        Evaluate fitness of each schedule
        Select top-performing schedules
        Perform crossover to generate new schedules
        Apply mutation to introduce variability
    End For
    Return best schedule found
End Procedure
```

Key Terms:

- **Fitness: Measure of how well a schedule meets objectives.**
- **Crossover: Combines two schedules to form a new one.**
- **Mutation: Introduces small changes to avoid local optima.**

Other Metaheuristics



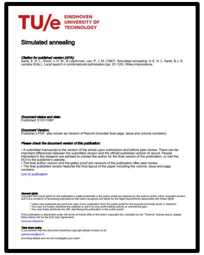
Simulated Annealing (SA):

Inspired by the annealing process in metallurgy.

Process:

- Starts with a **high "temperature"** (randomness).
- **Gradually cools**, refining solutions over time.

Use Case: Task scheduling for **makespan and cost minimization** and **resource load balance**


TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Simulated annealing

Aarts, E., Korst, J., & Michiels, W. (2005). Simulated annealing. *Search methodologies: introductory tutorials in optimization and decision support techniques*, 187-210.

Benefits of AI-Driven Scheduling

Adaptive:

- Can respond dynamically to changes in workload or resource availability.

Efficient for Large-Scale Problems:

- Handles high-dimensional search spaces effectively.

Multi-Objective Capabilities:

- Balances trade-offs like speed, cost, and energy consumption.

Example:

- Using **PSO** to balance CPU usage across multiple virtual machines.

Challenges in AI-Driven Scheduling

Computational Overhead:

- Metaheuristics may require significant computing power, especially for **real-time scheduling**.

Parameter Tuning:

- Algorithms like GA and PSO require **careful tuning of parameters** (e.g., mutation rate, swarm size).

Convergence Issues:

- Risk of "**getting stuck"** in **local optima.**

Example:

- GA might produce suboptimal task allocations if mutation is too low.

Real-World Use Cases

IoT Data Processing:

- **Example:** Real-time sensor data analysis in a smart factory.

Big Data Analytics:

- **HPC**: Slurm and Apache Hadoop Yarn
- **Cloud:** AWS Fargate (Serverless + Containers) & Google Cloud Scheduler.

Real-World Use Cases

Machine Learning:

- **Training and deployment** of large-scale ML models.
- **Example:** Google TPUs for deep learning workloads.

Streaming and Gaming Services:

- **Example:** FORTNITE scaling resources to handle high-demand content.

Hands-on Activity

Hands-On Activity

Simulating Scheduling Algorithms

- **Objective:** Implement and compare FCFS, Round Robin, MIN-MIN - Genetic and PSO.
- **Instructions:**
  - Use the provided Jupyter Notebook (Colab - Python)
  - Experiment with different task loads and parameters (e.g. quantum values, random).
  - Analyze execution times for each algorithm.

Questions for Discussion:

1. Which algorithm performs better in terms of **total time**?
2. How does the **quantum** value affect Round Robin's performance?
3. Which **factors are critical** when choosing a scheduling algorithm?

Scheduling Example – Computing Power Allocation

Optimizing Task Scheduling in a High-Performance Computing (HPC) Environment

Scenario:

A research institution runs simulations that require a large amount of computing power across multiple servers. Each simulation task has varying computational requirements and deadlines. Efficient scheduling is critical to ensure optimal usage of computing resources while meeting task deadlines.

Details

- **Resources:**
  - 4 servers with different units of computing power:
    - **Server A:** 10 units of computing power.
    - **Server B:** 8 units of computing power.
    - **Server C:** 6 units of computing power.
    - **Server D:** 5 units of computing power.
- **Tasks:**
  - **Task 1**: Requires 20 units of computing power, deadline = 4 hours.
  - **Task 2**: Requires 10 units of computing power, deadline = 2 hours.
  - **Task 3**: Requires 15 units of computing power, deadline = 3 hours.
  - **Task 4:** Requires 5 units of computing power, deadline = 1 hour.

Challenge:

How can we assign tasks to servers to:

1. **Meet** deadlines.
2. **Minimize** the total execution time.
3. **Balance** the computational load across servers.

Scheduling Example – Computing Power Allocation

Solution Using Scheduling

Step 1: Identify Available Resources

Each server contributes a fraction of the required computing power.

- Server A: Contributes 10 units/hour.
- Server B: Contributes 8 units/hour.
- Server C: Contributes 6 units/hour.
- Server D: Contributes 5 units/hour.

Step 2: Select Scheduling Algorithm

1. **Round Robin:**
   Assign tasks to servers in a cyclic order until the task's computing requirements are met.
2. **Min-Min:**
   Assign the shortest task (in terms of computing power needed) to the server with the most availability.

Scheduling Example – Computing Power Allocation

Result with Min-Min Scheduling

Task Assignment:

- **Task 1 (20 units):** Assigned to Server A and Server B (10 units/hour each) → Completed in 2 hours.
- **Task 2 (10 units):** Assigned to Server C (6 units/hour) and Server D (4 units/hour) → Completed in 1.67 hours.
- **Task 3 (15 units):** Assigned to Server A (10 units/hour) and Server B (5 units/hour) → Completed in 1.5 hours.
- **Task 4 (5 units):** Assigned to Server D (5 units/hour) → Completed in 1 hour.

Benefits

1. **Efficiency:**
   Resources are utilized optimally, with no idle servers.
2. **Deadline Compliance:**
   Tasks are completed within their respective deadlines.
3. **Balanced Load:**
   The computational load is distributed across servers, preventing bottlenecks.

Summary

Final Remarks

- Resource management and scheduling are crucial for optimizing performance, cost, and efficiency in cloud environments.
- Each algorithm has strengths and weaknesses depending on the use case.
- Trends like AI-driven scheduling and energy optimization are shaping the future of cloud computing.

Vielen Dank!